9.3

IBM MQ Skorowidz tworzenia aplikacji

Uwaga

Przed skorzystaniem z niniejszych informacji oraz produktu, którego one dotyczą, należy zapoznać się z informacjami zamieszczonymi w sekcji “Uwagi” na stronie 2267.

Niniejsze wydanie publikacji dotyczy wersji 9, wydania 3 produktu IBM® MQ oraz wszystkich jego późniejszych wydań i modyfikacji, aż do odwołania w nowych wydaniach publikacji.

Spis treści

Informacje dodatkowe o tworzeniu aplikacji

Odsyłacze znajdujące się w tej sekcji ułatwiają tworzenie aplikacji IBM MQ .

- “Odwołania do aplikacji MQI” na stronie 7
- IBM i “IBM i Skorowidz programistyczny aplikacji (ILE/RPG)” na stronie 1023
- IBM i “Konwersja danych w systemie IBM i” na stronie 1482
- “Informacje dodatkowe o wyjściach użytkownika, wyjściach funkcji API i instalowalnych usługach” na stronie 1502
- “Klasy i interfejsy IBM MQ .NET” na stronie 1775
- “Klasy IBM MQ C++” na stronie 1839
- “Właściwości obiektów IBM MQ classes for JMS” na stronie 1952
- “Informacje dodatkowe dotyczące przesyłania komunikatów REST API” na stronie 2232

Zadania pokrewne

Projektowanie aplikacji

Odsyłacze pokrewne

Klasy IBM MQ dla bibliotek Java

Klasy produktu IBM MQ dla usługi JMS

Odwołania do aplikacji MQI

Odsyłacze znajdujące się w tej sekcji ułatwiają tworzenie aplikacji interfejsu kolejki komunikatów (Message Queue Interface-MQI).

- “Przykłady kodu” na stronie 8
- “Stałe” na stronie 61
- “Typy danych używane w MQI” na stronie 236
- “Wywołania funkcji” na stronie 644
- “Atrybuty obiektów” na stronie 824
- “Kody powrotu” na stronie 903
- “Reguły sprawdzania poprawności opcji MQI” na stronie 904
- “Kodowania maszynowe” na stronie 930
- “Opcje raportu i flagi komunikatów” na stronie 933
- “Wyjście konwersji danych” na stronie 937
- “Właściwości określone jako elementy MQRFH2” na stronie 961
- “konwersja stron kodowych” na stronie 970

Pojęcia pokrewne

“Informacje dodatkowe o wyjściach użytkownika, wyjściach funkcji API i instalowalnych usługach” na stronie 1502

Informacje znajdujące się w tej sekcji są pomocne podczas tworzenia programów zewnętrznych, programów zewnętrznych funkcji API i aplikacji usług, które można zainstalować:

Zadania pokrewne

Projektowanie aplikacji

Odsyłacze pokrewne

“Klasy i interfejsy IBM MQ .NET” na stronie 1775

Klasy i interfejsy IBM MQ .NET są wymienione w kolejności alfabetycznej. Opisano właściwości, metody i konstruktory.

“Klasy IBM MQ C++” na stronie 1839
Klasy języka C++ IBM MQ hermetyzują interfejs MQI (Message Queue Interface) systemu IBM MQ . Istnieje pojedynczy plik nagłówkowy C + +, **imqi.hpp**, który obejmuje wszystkie te klasy.

Biblioteki IBM MQ Classes for Java

IBM MQ Klasy w systemie JMS

Przykłady kodu

Informacje uzupełniające w tej sekcji służą do wykonywania zadań, które zaspokajają potrzeby biznesowe.

Przykłady dla języka C

Ta kolekcja tematów jest najczęściej pobierana z przykładowych aplikacji IBM MQ for z/OS . Mają one zastosowanie do wszystkich platform, z wyjątkiem przypadków, w których zaznaczono inaczej.

Nawiązywanie połączenia z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQCONN w celu połączenia programu z menedżerem kolejek w zadaniu wsadowym produktu z/OS .

Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BCA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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;
      }
      .
      .
      .
}
```

Rozłączanie z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQDISC do rozłączenia programu z menedżerem kolejek w zadaniu wsadowym z programu z/OS .

Zmienne używane w tym wyodrębnieniu kodu są zmiennymi, które zostały ustawione w pliku “Nawiązywanie połączenia z menedżerem kolejek” na stronie 8. Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BCA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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;
      }
⋮
```

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do utworzenia kolejki dynamicznej.

Ten ekstrakt jest pobierany z przykładowej aplikacji menedżera poczty (program CSQ4TCD1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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;
}
.
```

Otwieranie istniejącej kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do otwarcia kolejki, która została już zdefiniowana.

Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BCA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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 */
```

Zamykanie kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQCLOSE do zamknięcia kolejki.

Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BCA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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;
      }
.
```

Umieszczanie komunikatu przy użyciu MQPUT

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT do umieszczenia komunikatu w kolejce.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ. Nazwy i położenia przykładowych aplikacji zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem z/OS) z/OS i Przykładowe programy dla produktu IBM MQ for z/OS .

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

    /*-------------------------------*/
    /*  Set up MD structure.         */
    /*-------------------------------*/
    memset(&MsgDesc, '\0', sizeof(MsgDesc));
    memcpy(MsgDesc.StrucId, MQMD_STRUC_ID,
           sizeof(MsgDesc.StrucId));
     MsgDesc.Version     =  MQMD_VERSION_1;
     MsgDesc.Expiry      =  MQEI_UNLIMITED;
     MsgDesc.Report      =  MQRO_NONE;
     MsgDesc.MsgType     =  MQMT_DATAGRAM;
```

```
    MsgDesc.Priority    = 1;
    MsgDesc.Persistence = MQPER_PERSISTENT;
    memset(MsgDesc.ReplyToQ,
           '\0',
           sizeof(MsgDesc.ReplyToQ));
    /*-----------------------------------------------------*/
    /*  Put the message.                                   */
    /*-----------------------------------------------------*/
    MQPUT(Hconn, Hobj, &MsgDesc, &PutMsgOpts,
          sizeof(message_buffer), message_buffer,
          &CompCode, &Reason);
```

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

Umieszczanie komunikatu przy użyciu komendy MQPUT1

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT1 do otwarcia kolejki, umieszczenia pojedynczego komunikatu w kolejce, a następnie zamknięcia kolejki.

Ten ekstrakt jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CCB5) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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;
   }
   .
   .
```

pobieranie komunikatu

W tym przykładzie przedstawiono sposób użycia wywołania MQGET do usunięcia komunikatu z kolejki.

Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BCA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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 */
```

Pobieranie komunikatu przy użyciu opcji wait

W tym przykładzie przedstawiono sposób użycia opcji wait wywołania MQGET.

Ten kod akceptuje obcięte komunikaty. Ten ekstrakt jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CCB5) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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();
         }
         .
```

Pobieranie komunikatu przy użyciu sygnalizacji

Sygnalizacja jest dostępna tylko w produkcie IBM MQ for z/OS .

W tym przykładzie przedstawiono sposób użycia wywołania MQGET do ustawienia sygnału w taki sposób, aby użytkownik był powiadamiany o nadejściu odpowiedniego komunikatu do kolejki. Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

```
 /*--------------------------*/
 /*  Set up MD structure.    */
 /*--------------------------*/
 memset(&MsgDesc,'\0',sizeof(MsgDesc));
 memcpy(MsgDesc.StrucId, MQMD_STRUC_ID,
        sizeof(MsgDesc.StrucId);
 MsgDesc.Version  = MQMD_VERSION_1;
 MsgDesc.Report   = MQRO_NONE;
 memcpy(MsgDesc.MsgId,MQMI_NONE,
        sizeof(MsgDesc.MsgId));
 memcpy(MsgDesc.CorrelId,MQCI_NONE,
        sizeof(MsgDesc.CorrelId));
```

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

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

```
if (signal_sw == 1)
```

```
       {
        endloop = 0;
        do
          {
           EXEC CICS DELAY FOR HOURS(0) MINUTES(0) SECONDS(1);
           work_ecb = q_ecb & mask;
           switch (work_ecb)
               {
                case (MQEC_MSG_ARRIVED):
                    endloop = 1;
                    mqgmo_options = MQGMO_NO_WAIT;
                    MQGET(Hconn, Hobj, &MsgDesc, &GetMsgOpts,
                          BufferLength, message_buffer,
                          &DataLength, &CompCode, &Reason);
                    if (CompCode != MQCC_OK)
                        ;          /* Perform error processing. */
                    break;
                case (MQEC_WAIT_INTERVAL_EXPIRED):
                case (MQEC_WAIT_CANCELED):
                    endloop = 1;
                    break;
                default:
                    break;
               }
          } while (endloop == 0);
        }
     return;
}
```

Uzyskiwanie informacji o atrybutach obiektu

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ do uzyskania informacji o atrybutach kolejki.

Ten fragment jest pobierany z przykładowej aplikacji Atrybuty kolejki (program CSQ4CCC1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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 */
```

Ustawianie atrybutów kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQSET do zmiany atrybutów kolejki.

Ten fragment jest pobierany z przykładowej aplikacji Atrybuty kolejki (program CSQ4CCC1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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 */
```

Pobieranie informacji o statusie za pomocą komendy MQSTAT

W tym przykładzie przedstawiono sposób wywołania asynchronicznej operacji MQPUT i pobrania informacji o statusie za pomocą komendy MQSTAT.

Ten fragment jest pobierany z przykładowej aplikacji MQSTAT (program amqsapt0) dostarczane z systemami IBM MQ for Windows . Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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);
}
```

Przykłady w języku COBOL

Ta kolekcja tematów jest pobierana z przykładowych aplikacji IBM MQ for z/OS . Mają one zastosowanie do wszystkich platform, z wyjątkiem przypadków, w których zaznaczono inaczej.

Nawiązywanie połączenia z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQCONN w celu połączenia programu z menedżerem kolejek w zadaniu wsadowym produktu z/OS .

Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BVA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
     ⋮
```

Rozłączanie z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQDISC do rozłączenia programu z menedżerem kolejek w zadaniu wsadowym z programu z/OS .

Zmienne używane w tym wyodrębnieniu kodu są zmiennymi, które zostały ustawione w pliku “Nawiązywanie połączenia z menedżerem kolejek” na stronie 23. Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BVA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
⋮
```

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do utworzenia kolejki dynamicznej.

Ten fragment jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CVB1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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
*
```

Otwieranie istniejącej kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do otwarcia istniejącej kolejki.

Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BVA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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
```

Zamykanie kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQCLOSE.

Zmienne używane w tym wyodrębnieniu kodu są zmiennymi, które zostały ustawione w pliku “Nawiązywanie połączenia z menedżerem kolejek” na stronie 23. Ten ekstrakt jest pobierany z przykładowej aplikacji przeglądania (program CSQ4BVA1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
*
```

Umieszczanie komunikatu przy użyciu MQPUT

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT z użyciem kontekstu.

Ten fragment jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CVB1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
```

Umieszczanie komunikatu przy użyciu komendy MQPUT1

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT1 .

Ten fragment jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CVB5) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
*
```

pobieranie komunikatu

W tym przykładzie przedstawiono sposób użycia wywołania MQGET do usunięcia komunikatu z kolejki.

Ten fragment jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CVB1) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
```

Pobieranie komunikatu przy użyciu opcji wait

W tym przykładzie przedstawiono sposób użycia wywołania MQGET z opcją wait i akceptowania obciętych komunikatów.

Ten fragment jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CVB5) dostarczanej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
⋮
```

Pobieranie komunikatu przy użyciu sygnalizacji

W tym przykładzie przedstawiono sposób użycia wywołania MQGET z sygnalizacją. Ten fragment jest pobierany z przykładowej aplikacji Credit Check (program CSQ4CVB2) dostarczanej z programem IBM MQ for z/OS.

Sygnalizacja jest dostępna tylko w produkcie IBM MQ for z/OS .

```
⋮
* ----------------------------------------------------------*
 WORKING-STORAGE SECTION.
* ----------------------------------------------------------*
*
*    W00 - General work fields
 ⋮
 01  W00-WAIT-INTERVAL    PIC S9(09) BINARY VALUE 30000.
```

```
*
*    W03 - MQM API fields
*
 01  W03-HCONN            PIC S9(9) BINARY VALUE ZERO.
 01  W03-HOBJ-REPLYQ      PIC S9(9) BINARY.
 01  W03-COMPCODE         PIC S9(9) BINARY.
 01  W03-REASON           PIC S9(9) BINARY.
 01  W03-DATALEN          PIC S9(9) BINARY.
 01  W03-BUFFLEN          PIC S9(9) BINARY.
 ⋮
 01  W03-GET-BUFFER.
     05 W03-CSQ4BQRM.
     COPY CSQ4VB4.
*
     05 W03-CSQ4BIIM REDEFINES W03-CSQ4BQRM.
     COPY CSQ4VB1.
*
     05 W03-CSQ4BPGM REDEFINES W03-CSQ4BIIM.
     COPY CSQ4VB5.
     ⋮
*    API control blocks
*
 01  MQM-MESSAGE-DESCRIPTOR.
     COPY CMQMDV.
 01  MQM-GET-MESSAGE-OPTIONS.
     COPY CMQGMOV.
     ⋮
*    MQV contains constants (for filling in the
*    control blocks) and return codes (for testing
*    the result of a call).
*
 01  MQM-MQV.
 COPY CMQV SUPPRESS.
* ------------------------------------------------------*
 LINKAGE SECTION.
* ------------------------------------------------------*
 01  L01-ECB-ADDR-LIST.
     05  L01-ECB-ADDR1        POINTER.
     05  L01-ECB-ADDR2        POINTER.
```

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

```
*
      WHEN OTHER
          MOVE 'MQGET SIGNAL'  TO M02-OPERATION
          MOVE MQOD-OBJECTNAME TO M02-OBJECTNAME
          PERFORM RECORD-CALL-ERROR
          MOVE W06-CALL-ERROR  TO W06-CALL-STATUS
  END-EVALUATE.
*
 PROCESS-SIGNAL-ACCEPTED-EXIT.
*    Return to performing section
     EXIT.
     EJECT
*
```

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

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

```
*
 REPLYQ-GETSIGNAL-EXIT.
*
*    Return to performing section.
*
     EXIT.
     EJECT
*
     ⋮
```

Uzyskiwanie informacji o atrybutach obiektu

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ do uzyskania informacji o atrybutach kolejki.

Ten ekstrakt jest pobierany z przykładowej aplikacji Queue Attributes (program CSQ4CVC1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
        ⋮
```

Ustawianie atrybutów kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQSET do zmiany atrybutów kolejki.

Ten ekstrakt jest pobierany z przykładowej aplikacji Queue Attributes (program CSQ4CVC1) dostarczonej z produktem IBM MQ for z/OS. Nazwy i położenia przykładowych aplikacji na innych platformach zawiera sekcja Przykładowe programy proceduralne (platformy z wyjątkiem 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.
```

Przykłady języka asemblera systemu System/390

Ta kolekcja tematów jest najczęściej pobierana z przykładowych aplikacji IBM MQ for z/OS .

Nawiązywanie połączenia z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQCONN w celu połączenia programu z menedżerem kolejek w zadaniu wsadowym produktu z/OS .

Ten fragment jest pobierany z przykładowego programu przeglądania (CSQ4BAA1) dostarczanego z produktem IBM MQ for z/OS.

```
⋮
WORKAREA DSECT
*
PARMLIST CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
COMPCODE DS    F              Completion code
REASON   DS    F              Reason code
HCONN    DS    F              Connection handle
         ORG
PARMADDR DS    F              Address of parm field
PARMLEN  DS    H              Length of parm field
*
MQMNAME  DS    CL48           Queue manager name
*
*
************************************************************
*  SECTION NAME : MAINPARM                                 *
************************************************************
MAINPARM DS    0H
         MVI   MQMNAME,X'40'
         MVC   MQMNAME+1(L'MQMNAME-1),MQMNAME
*
* Space out first byte and initialize
*
*
```

```
* Code to address and verify parameters passed omitted
*
*
PARM1MVE DS    0H
         SR    R1,R3           Length of data
         LA    R4,MQMNAME      Address for target
         BCTR  R1,R0           Reduce for execute
         EX    R1,MOVEPARM     Move the data
*
************************************************************
* EXECUTES                                                 *
************************************************************
MOVEPARM MVC   0(*-*,R4),0(R3)
*
         EJECT
```

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

Rozłączanie z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQDISC do rozłączenia programu z menedżerem kolejek w zadaniu wsadowym z programu z/OS .

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

```
BADCALL  DS    0H
⋮
*                   CONSTANTS
*
         CMQA
*
```

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

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do utworzenia kolejki dynamicznej.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

```
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L  LIST FORM
                                                 OF CALL
*                                                 MACRO
         ⋮
         END
```

Otwieranie istniejącej kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do otwarcia kolejki, która została już zdefiniowana.

Przedstawia on sposób określania dwóch opcji. Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

Zamykanie kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQCLOSE do zamknięcia kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

```
⋮
*
* ISSUE MQI CLOSE REQUEST USING REENTRANT FROM OF
* CALL MACRO
*
*        HCONN WAS SET BY A PREVIOUS MQCONN REQUEST
*        HOBJ  WAS SET BY A PREVIOUS MQOPEN REQUEST
*        R5 = WORK REGISTER
*
CLOSE    DS    0H
         LA    R5,MQCO_NONE        NO SPECIAL CLOSE OPTIONS
         ST    R5,OPTIONS              ARE REQUIRED.
*
         CALL  MQCLOSE,                       X
               (HCONN,                        X
               HOBJ,                          X
               OPTIONS,                       X
               COMPCODE,                      X
               REASON),                       X
               VL,MF=(E,CALLLST)
*
         LA    R5,MQCC_OK
         C     R5,COMPCODE
         BNE   BADCALL
*
         ⋮
BADCALL  DS    0H
         ⋮
*                   CONSTANTS
*
         CMQA
*
*        WORKING STORAGE (REENTRANT)
*
WEG4     DSECT
*
CALLLST  CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
HCONN    DS   F
HOBJ     DS   F
OPTIONS  DS   F
COMPCODE DS   F
REASON   DS   F
*
*
LEG4     EQU  *-WKEG4
         END
```

Umieszczanie komunikatu przy użyciu MQPUT

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT do umieszczenia komunikatu w kolejce.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

Umieszczanie komunikatu przy użyciu komendy MQPUT1

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT1 do otwarcia kolejki, umieszczenia pojedynczego komunikatu w kolejce, a następnie zamknięcia kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

```
⋮
*
*     CONNECT TO QUEUE MANAGER
*
CONN    DS  0H
⋮
*
```

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

```
*
         MVC  WPMO_AREA,MQPMO_AREA       INITIALIZE WORKING MQPMO
*

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

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

```
         ⋮
         END
```

pobieranie komunikatu

W tym przykładzie przedstawiono sposób użycia wywołania MQGET do usunięcia komunikatu z kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

```
*
*      CONSTANTS
*
         CMQMDA DSECT=NO,LIST=YES
         CMQGMOA DSECT=NO,LIST=YES
         CMQA
*
*      WORKING STORAGE DSECT
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
```

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

Pobieranie komunikatu przy użyciu opcji wait

W tym przykładzie przedstawiono sposób użycia opcji wait wywołania MQGET.

Ten kod akceptuje obcięte komunikaty. Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

```
                                  IS IT DUE TO AN EMPTY
  C   R5,REASON                   QUEUE?
  BE  NOMSG                       YES, SO HANDLE THE ERROR
  B   BADCALL                     NO, SO GO TO ERROR ROUTINE
*
CHECK_W  DS  0H
         LA  R5,MQRC_TRUNCATED_MSG_ACCEPTED   IS THIS A
                                              TRUNCATED
         C   R5,REASON                        MESSAGE?
         BE  GETOK                   YES, SO GO AND PROCESS.
         B   BADCALL                 NO, SOME OTHER WARNING
*
NOMSG    DS  0H
         ⋮
GETOK    DS  0H
         ⋮
```

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

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

Pobieranie komunikatu przy użyciu sygnalizacji

W tym przykładzie przedstawiono sposób użycia wywołania MQGET do ustawienia sygnału w taki sposób, aby użytkownik był powiadamiany o nadejściu odpowiedniego komunikatu do kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

```
⋮
*
*     CONNECT TO QUEUE MANAGER
*
CONN    DS  0H
        ⋮
*
*     OPEN A QUEUE FOR GET
*
OPEN    DS  0H
        ⋮
*
*     R4,R5,R6,R7 = WORK REGISTER.
*
GET  DS  0H
     LA   R4,MQMD               SET UP ADDRESSES AND
```

```
     LA   R5,MQMD_LENGTH         LENGTH FOR USE BY MVCL
     LA   R6,WMD                 INSTRUCTION, AS MQMD IS
     LA   R7,WMD_LENGTH          OVER 256 BYES LONG.
     MVCL R6,R4                  INITIALIZE WORKING VERSION
*                                   OF MESSAGE DESCRIPTOR
```

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

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

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

```
BADCALL DS  0H
        ⋮
*
*      CONSTANTS
*
         CMQMDA DSECT=NO,LIST=YES
         CMQGMOA DSECT=NO,LIST=YES
         CMQA
*
FIVE_MINUTES DC F'300000'        GET SIGNAL INTERVAL
*
*      WORKING STORAGE DSECT
*
WORKSTG  DSECT
*
COMPCODE DS F
REASON   DS F
BUFFLEN  DS F
DATALEN  DS F
OPTIONS  DS F
HCONN    DS F
HOBJ     DS F
SIG_ECB  DS F
```

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

Uzyskiwanie informacji o atrybutach kolejki i ustawianie tych atrybutów

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ w celu uzyskania informacji o atrybutach kolejki oraz użycia wywołania MQSET w celu zmiany atrybutów kolejki.

Ten fragment jest pobierany z przykładowej aplikacji Atrybuty kolejki (program CSQ4CAC1) dostarczanej z produktem IBM MQ for z/OS.

```
⋮
DFHEISTG DSECT
         ⋮
OBJDESC  CMQODA LIST=YES   Working object descriptor
*
SELECTORCOUNT   DS F       Number of selectors
INTATTRCOUNT    DS F       Number of integer attributes
CHARATTRLENGTH  DS F       char attributes length
CHARATTRS       DS C       Area for char attributes
*
OPTIONS  DS   F            Command options
HCONN    DS   F            Handle of connection
HOBJ     DS   F            Handle of object
COMPCODE DS   F            Completion code
REASON   DS   F            Reason code
SELECTOR DS   2F           Array of selectors
INTATTRS DS   2F           Array of integer attributes
         ⋮
OBJECT   DS   CL(MQ_Q_NAME_LENGTH)   Name of queue
         ⋮
CALLLIST CALL ,(0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
************************************************************
*                   PROGRAM EXECUTION STARTS HERE        *
⋮
CSQ4CAC1 DFHEIENT CODEREG=(R3),DATAREG=(R13)
         ⋮
*        Initialize the variables for the set call
*
         SR   R0,R0               Clear register zero
         ST   R0,CHARATTRLENGTH   Set char length to zero
         LA   R0,2                Load to set
         ST   R0,SELECTORCOUNT    selectors add
```

```
         ST   R0,INTATTRCOUNT     integer attributes
*
         LA   R0,MQIA_INHIBIT_GET Load q attribute selector
         ST   R0,SELECTOR+0       Place in field
         LA   R0,MQIA_INHIBIT_PUT Load q attribute selector
         ST   R0,SELECTOR+4       Place in field
*
UPDTEST  DS   0H
         CLC  ACTION,CINHIB       Are we inhibiting?
         BE   UPDINHBT            Yes branch to section
*
         CLC  ACTION,CALLOW       Are we allowing?
         BE   UPDALLOW            Yes branch to section
*
         MVC  M00_MSG,M01_MSG1    Invalid request
         BR   R6                  Return to caller
*
```

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

```
*        Initialize the variables for the inquire call
*
         SR   R0,R0               Clear register zero
         ST   R0,CHARATTRLENGTH   Set char length to zero
         LA   R0,2                Load to set
         ST   R0,SELECTORCOUNT    selectors add
         ST   R0,INTATTRCOUNT     integer attributes
*
         LA   R0,MQIA_INHIBIT_GET Load attribute value
         ST   R0,SELECTOR+0       Place in field
         LA   R0,MQIA_INHIBIT_PUT Load attribute value
         ST   R0,SELECTOR+4       Place in field
         CALL MQINQ,                          C
               (HCONN,                        C
               HOBJ,                          C
               SELECTORCOUNT,                 C
               SELECTOR,                      C
               INTATTRCOUNT,                  C
               INTATTRS,                      C
```

```
            CHARATTRLENGTH,               C
            CHARATTRS,                    C
            COMPCODE,                     C
            REASON),                      C
            VL,MF=(E,CALLLIST)
      LA    R0,MQCC_OK        Load expected compcode
      C     R0,COMPCODE       Was inquire successful?
      ⋮
```

Przykłady w języku PL/I

Użycie języka PL/I jest obsługiwane tylko przez język z/OS . Ta kolekcja tematów przedstawia techniki wykorzystujące przykłady w języku PL/I.

Nawiązywanie połączenia z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQCONN w celu połączenia programu z menedżerem kolejek w zadaniu wsadowym produktu z/OS .

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

Rozłączanie z menedżerem kolejek

W tym przykładzie przedstawiono sposób użycia wywołania MQDISC do rozłączenia programu z menedżerem kolejek w zadaniu wsadowym z programu z/OS .

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

```
%INCLUDE SYSLIB(CMQP);
%INCLUDE SYSLIB(CMQEPP);
```

```
     :
     /*****************************************************/
     /* WORKING STORAGE DECLARATIONS                      */
     /*****************************************************/
     DCL COMPCODE                  BINARY FIXED (31);
     DCL REASON                    BINARY FIXED (31);
     DCL HCONN                     BINARY FIXED (31);
 ⋮
     /*****************************************************/
     /* DISCONNECT FROM THE QUEUE MANAGER                 */
     /*****************************************************/
     CALL MQDISC (HCONN,       /* CONNECTION HANDLE      */
                  COMPCODE,    /* COMPLETION CODE        */
                  REASON);     /* REASON CODE            */

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

Tworzenie kolejki dynamicznej

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do utworzenia kolejki dynamicznej.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

```
          CALL ERROR_ROUTINE;
        END;
        ELSE
          DYNAMIC_QUEUE_NAME = LMQOD_OBJECTNAME;
```

Otwieranie istniejącej kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQOPEN do otwarcia istniejącej kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

Zamykanie kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQCLOSE.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

```
/*********************************************************/
/* CLOSE QUEUE                                           */
/*********************************************************/
   CALL MQCLOSE (HCONN,      /* CONNECTION HANDLE        */
                 HOBJ,       /* OBJECT HANDLE            */
                 OPTIONS,    /* CLOSE OPTIONS            */
                 COMPCODE,   /* COMPLETION CODE          */
                 REASON);    /* REASON CODE              */

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

Umieszczanie komunikatu przy użyciu MQPUT

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT z użyciem kontekstu.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

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

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

Umieszczanie komunikatu przy użyciu komendy MQPUT1

W tym przykładzie przedstawiono sposób użycia wywołania MQPUT1 .

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

/*************************************************************/
/* SET UP LENGTH OF MESSAGE BUFFER AND THE MESSAGE    */
```

```
/*************************************************************/
BUFFLEN = LENGTH(BUFFER);
BUFFER = PL1_TEST_MESSAGE;

CALL MQPUT1 (HCONN,
LMQOD,
LMQMD,
LMQPMO,
BUFFLEN,
BUFFER,
COMPCODE,
REASON);

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

pobieranie komunikatu

W tym przykładzie przedstawiono sposób użycia wywołania MQGET do usunięcia komunikatu z kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

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

     CALL MQGET (HCONN,
```

```
                HOBJ,
                LMQMD,
                LMQGMO,
                BUFFERLEN,
                BUFFER,
                DATALEN,
                COMPCODE,
                REASON);

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

Pobieranie komunikatu przy użyciu opcji wait

W tym przykładzie przedstawiono sposób użycia wywołania MQGET z opcją wait i akceptowania obciętych komunikatów.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

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

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

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

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

Pobieranie komunikatu przy użyciu sygnalizacji

Wyodrębniony kod demonstrujący sposób użycia wywołania MQGET z sygnalizacją.

Sygnalizacja jest dostępna tylko w produkcie IBM MQ for z/OS .

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

```
/*********************************************************/
/* SET UP MESSAGE DESCRIPTOR AS REQUIRED.                */
/* MSGID AND CORRELID IN MQMD SET TO NULLS SO FIRST      */
/* AVAILABLE MESSAGE WILL BE RETRIEVED.                  */
/*********************************************************/
     LMQMD.MSGID = MQMI_NONE;
     LMQMD.CORRELID = MQCI_NONE;
/*********************************************************/
/* SET UP GET MESSAGE OPTIONS AS REQUIRED.               */
/* WAIT INTERVAL SET TO ONE MINUTE.                      */
/*********************************************************/
     LMQGMO.OPTIONS = MQGMO_SET_SIGNAL +
                      MQGMO_NO_SYNCPOINT;
     LMQGMO.WAITINTERVAL=60000;
     LMQGMO.SIGNAL1 = ADDR(ECB_FIXED);
```

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

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

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

```
 DO_WORK: PROC;
     ⋮
     IF ECB_POSTED
        THEN DO;
          SELECT(ECB_CODE);
            WHEN(MQEC_MSG_ARRIVED) DO;
              ⋮
              CALL GET_MSG;
              ⋮
            END;
            WHEN(MQEC_WAIT_INTERVAL_EXPIRED) DO;
              ⋮
              CALL NO_MSG;
              ⋮
            END;
            OTHERWISE DO;        /* FAILURE CASE */
/*********************************************************/
```

```
/* ISSUE AN ERROR MESSAGE SHOWING THE COMPLETION CODE  */
/* AND THE REASON CODE.                                */
/*******************************************************/
              ⋮
              CALL ERROR_ROUTINE;
              ⋮
           END;

          END;

       END;
     ⋮
 END DO_WORK;

 GET_MSG: PROC;
```

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

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

 END GET_MSG;

 NO_MSG: PROC;
 ⋮
 END NO_MSG;
```

Uzyskiwanie informacji o atrybutach obiektu

W tym przykładzie przedstawiono sposób użycia wywołania MQINQ do uzyskania informacji o atrybutach kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

     SELECTORS(01) = MQIA_CURRENT_Q_DEPTH;

     SELECTORCOUNT  = 1;
     INTATTRCOUNT   = 1;
```

```
     CHARATTRLENGTH = 0;
/*******************************************************/
/*                                                     */
/* HCONN WAS SET BY PREVIOUS MQCONN REQUEST.           */
/* HOBJ WAS SET BY PREVIOUS MQOPEN REQUEST.            */
/*                                                     */
/*******************************************************/
     CALL MQINQ (HCONN,
                 HOBJ,
                 SELECTORCOUNT,
                 SELECTORS,
                 INTATTRCOUNT,
                 INTATTRS,
                 CHARATTRLENGTH,
                 CHARATTRS,
                 COMPCODE,
                 REASON);
```

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

Ustawianie atrybutów kolejki

W tym przykładzie przedstawiono sposób użycia wywołania MQSET do zmiany atrybutów kolejki.

Ten ekstrakt nie jest pobierany z przykładowych aplikacji dostarczonych z produktem IBM MQ.

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

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

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

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

     SELECTORCOUNT  = 2;
     INTATTRCOUNT   = 2;

     CHARATTRLENGTH = 0;
```

```
/*******************************************************/
/*                                                     */
```

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

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

Stałe

Informacje uzupełniające w tej sekcji służą do wykonywania zadań, które zaspokajają potrzeby biznesowe.

IBM MQ Pliki COPY, header, include i module

Te informacje są ogólnymi informacjami o interfejsie programistycznym.

Ta sekcja zawiera informacje pomocne podczas używania interfejsu MQI w różnych językach programowania.

Pliki nagłówkowe języka C

Pliki nagłówkowe są udostępniane w celu ułatwienia pisania aplikacji w języku C, które używają interfejsu MQI.

Pliki nagłówkowe w języku C zostały podsumowane w poniższej tabeli:

Tabela 1. Pliki nagłówkowe języka C-wywoływanie prototypów, typów danych, kodów powrotu, stałych i struktur

Nazwa pliku	Opis	IBM i	Systemy AIX and Linux®	Windows	z/OS
Wywoływanie prototypów, typów danych, kodów powrotu, stałych i struktur					
CMQC	Definicje MQI	C	C	C	C
CMQBC,	Definicje MQAI	C	C	C	
CMQEC	Definicja punktów wejścia interfejsu (obejmuje CMQC, CMQXC i CMQZC)		C	C	
CMQCFC	Definicje PCF	C	C	C	C
KPiB	Definicje publikowania/ subskrypcji	C	C	C	C
CMQXC	Definicje kanału i wyjścia	C	C	C	C
CMQZC	Definicje usług instalowalnych	C	C	C	

Tabela 1. Pliki nagłówkowe języka C-wywoływanie prototypów, typów danych, kodów powrotu, stałych i struktur (kontynuacja)					
Nazwa pliku	**Opis**	**IBM i**	**Systemy AIX and Linux®**	**Windows**	**z/OS**
Klucz: C= Pliki udostępnione					

Pliki COPY języka COBOL

Dostępne są różne pliki COPY, które ułatwiają pisanie aplikacji w języku COBOL korzystających z interfejsu MQI.

Tabela 2. Pliki kopii COBOL-kody powrotu, stałe i struktury					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Kody powrotu i stałe					
CMQx	Definicje MQI	V	V	V	V
CMQCFX	Definicje PCF	V	V	V	V
CMQPSx	Definicje publikowania/subskrypcji	V	V	V	V
CMQXx	Definicje kanału i wyjścia	V	V	V	V
Struktury					
CMQAIRX	MQAIR-rekord informacji uwierzytelniającej		L	L	
CMQBOX	MQBO-opcje początku	L	L	L	
CMQCDx	MQCD-definicja kanału	L	L	L	L
CMQCFBFx	MQCFBF-parametr filtru łańcucha bajtów PCF	L	L	L	L
CMQCFBSx	MQCFBS-parametr łańcucha bajtów PCF	L	L	L	L
CMQCFGRx	MQCFGR-parametr grupy PCF	L	L	L	L
CMQCFHx	MQCFH-nagłówek PCF	L	L	L	L
CMQCFIFx	MQCFIF-parametr filtru całkowitoliczbowego PCF	L	L	L	L
CMQCFILx	MQCFIL-parametr listy liczb całkowitych PCF	L	L	L	L
CMQCFINx	MQCFIN-parametr liczby całkowitej PCF	L	L	L	L
CMQCFSFx	MQCFSF-parametr filtru łańcucha PCF	L	L	L	L
CMQCFSLX	MQCFSL-parametr listy łańcuchów PCF	L	L	L	L
CMQCFSTx	MQCFST-parametr łańcucha PCF	L	L	L	L
CMQCFXLx	MQCFIL64 -parametr listy 64-bitowych liczb całkowitych PCF	L	L	L	L

Tabela 2. Pliki kopii COBOL-kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQCFXNx	MQCFIN64 -64-bitowy parametr całkowitoliczbowy PCF	L	L	L	L
CMQCHRVx	MQCHARV-łańcuch o zmiennej długości	L	L	L	L
CMQCIHx	MQCIH-nagłówek CICS bridge	L	L	L	L
CMQCNOx	MQCNO-opcje połączenia	L	L	L	L
CMQCSPx	MQCSP-parametry zabezpieczeń	L	L	L	L
CMQCXPx	MQCXP-parametry wyjścia kanału	L			L
Moduł CMQDHx	MQDH-nagłówek dystrybucji	L	L	L	L
CMQDLHx	MQDLH-nagłówek niedostarczonego komunikatu	L	L	L	L
CMQDXPx	MQDXP-parametry wyjścia konwersji danych	L		L	
CMQEPHx	MQEPH-wbudowany nagłówek PCF	L	L	L	L
CMQGMOx	MQGMO-opcje pobierania komunikatu	L	L	L	L
CMQIIHx	MQIIH-nagłówek informacji IMS	L	L	L	L
CMQMDx	MQMD-deskryptor komunikatu	L	L	L	L
CMQMD1x	MQMD1 -deskryptor komunikatu w wersji 1	L	L	L	L
CMQMD2x	MQMD2 -deskryptor komunikatu w wersji 2	L	L	L	L
CMQMDExName	MQMDE-rozszerzony deskryptor komunikatu	L	L	L	L
CMQODx	MQOD-deskryptor obiektu	L	L	L	L
CMQORx	MQOR-rekord obiektu	L	L	L	L
Komenda CMQPMOx	MQPMO-opcje umieszczania komunikatów	L	L	L	L
CMQRFHx	MQRFH-reguły i nagłówek formatowania	L	L	L	L
CMQRFH2x	MQRFH2 -reguły i nagłówek formatowania 2	L	L	L	L
CMQRMHx	MQRMH-nagłówek komunikatu referencyjnego	L	L	L	L
CMQRRx	MQRR-rekord odpowiedzi	L	L	L	
CMQSCOx	MQSCO-opcje konfiguracji TLS		L	L	
CMQTMX	MQTM-komunikat wyzwalacza	L		L	L

Tabela 2. Pliki kopii COBOL-kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQTMCx	MQTMC-znak komunikatu wyzwalacza	L	L		
CMQTMC2x	MQTMC2 -komunikat wyzwalacza 2-znak	L	L	L	L
CMQWIHx	MQWIH-nagłówek informacji o pracy	L	L	L	L
CMQXQHx	MQXQH-nagłówek kolejki transmisji	L	L	L	L
Klucz: • Pliki z podanymi wartościami początkowymi, x = V • Pliki bez podanych wartości początkowych, x = L					

z/OS *Pliki włączane języka PL/I*

Dla języka programowania PL/I udostępniono wiele plików INCLUDE. Te pliki są dostępne tylko w systemie z/OS .

Tabela 3. Pliki włączane w języku PL/I-typy danych, kody powrotu, stałe i struktury					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Typy danych, kody powrotu, stałe i struktury					
CMQP	Definicje MQI				P
CMQCFP	Definicje PCF				P
CMQEPP	Definicje punktów wejścia				P
Usługa CMQPSP	Definicje publikowania/ subskrypcji				P
CMQXP	Definicje kanału i wyjścia				P
Klucz: P= Plik udostępniony					

IBM i *Pliki kopii RPG*

Pliki RPG COPY są dostarczane dla języka programowania RPG. Te pliki są dostępne tylko w systemie IBM i.

Tabela 4. Kopiowanie plików RPG-kody powrotu, stałe i struktury					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Kody powrotu i stałe					
CMQx	Definicje MQI	G R			
CMQCFX	Definicje PCF	G			
CMQPSx	Definicje publikowania/subskrypcji	G			
CMQXx	Definicje kanału i wyjścia	G R			

Tabela 4. Kopiowanie plików RPG-kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Struktury					
CMQBOX	MQBO-opcje początku	G H			
CMQCDx	MQCD-definicja kanału	G H R			
CMQCFBFx	MQCFBF-parametr filtru łańcucha bajtów PCF	G H			
CMQCFBSx	MQCFBS-parametr łańcucha bajtów PCF	G H			
CMQCFGRx	MQCFGR-parametr grupy PCF	G H			
CMQCFHx	MQCFH-nagłówek PCF	G H			
CMQCFIFx	MQCFIF-parametr filtru całkowitoliczbowego PCF	G H			
CMQCFILx	MQCFIL-parametr listy liczb całkowitych PCF	G H			
CMQCFINx	MQCFIN-parametr liczby całkowitej PCF	G H			
CMQCFSFx	MQCFSF-parametr filtru łańcucha PCF	G H			
CMQCFSLX	MQCFSL-parametr listy łańcuchów PCF	G H			
CMQCFSTx	MQCFST-parametr łańcucha PCF	G H			
CMQCFXLx	MQCFIL64 -parametr listy 64-bitowych liczb całkowitych PCF	G H			
CMQCFXNx	MQCFIN64 -64-bitowy parametr całkowitoliczbowy PCF	G H			
CMQCHARVx	MQCHARV-łańcuch o zmiennej długości	G H			
CMQCIHx	MQCIH-nagłówek CICS bridge	G H			
CMQCNOx	MQCNO-opcje połączenia	G H			
CMQCSPx	MQCSP-parametry zabezpieczeń	G H			
CMQCXPx	MQCXP-parametry wyjścia kanału	G H R			
Moduł CMQDHx	MQDH-nagłówek dystrybucji	G H R			
CMQDLHx	MQDLH-nagłówek niedostarczonego komunikatu	G H R			
CMQDXPx	MQDXP-parametry wyjścia konwersji danych	G H R			
CMQEPHx	MQEPH-wbudowany nagłówek PCF	G H			
CMQGMOx	MQGMO-opcje pobierania komunikatu	G H R			

Tabela 4. Kopiowanie plików RPG-kody powrotu, stałe i struktury (kontynuacja)

Nazwa pliku	Opis	IBM i	AIX and Linux	Windows	z/OS
CMQIIHx	MQIIH-nagłówek informacji IMS	G H R			
CMQMDx	MQMD-deskryptor komunikatu	G H R			
CMQMD1x	MQMD1 -deskryptor komunikatu w wersji 1	G H R			
CMQMD2x	MQMD2 -deskryptor komunikatu w wersji 2	G H			
CMQMDExName	MQMDE-rozszerzony deskryptor komunikatu	G H R			
CMQODx	MQOD-deskryptor obiektu	G H R			
CMQORx	MQOR-rekord obiektu	G H R			
Komenda CMQPMOx	MQPMO-opcje umieszczania komunikatów	G H R			
CMQPXPx	MQPXP-Parametry wyjścia routingu publikowania/subskrypcji	G H			
CMQRFHx	MQRFH-reguły i nagłówek formatowania	G H			
CMQRFH2x	MQRFH2 -reguły i nagłówek formatowania 2	G H			
CMQRMHx	MQRMH-nagłówek komunikatu referencyjnego	G H R			
CMQRRx	MQRR-rekord odpowiedzi	G H R			
CMQTMX	MQTM-komunikat wyzwalacza	G H R			
CMQTMCx	MQTMC-znak komunikatu wyzwalacza	G H R			
CMQTMC2x	MQTMC2 -komunikat wyzwalacza 2-znak	G H R			
CMQWIHx	MQWIH-nagłówek informacji o pracy	G H			
CMQXQHx	MQXQH-nagłówek kolejki transmisji	G H R			

Klucz:

- Plik dla połączenia statycznego, zainicjowany, udostępniony x = G
- Plik dla połączenia statycznego, niezainicjowany, podany x = H
- Plik dla połączenia dynamicznego, zainicjowany, udostępniony, x = R

Windows *Pliki modułu Visual Basic*

Pliki nagłówkowe (lub pliki formularzy) są udostępniane w celu ułatwienia pisania aplikacji Visual Basic, które używają interfejsu MQI. Te pliki nagłówkowe są dostarczane tylko w wersjach 32-bitowych.

Tabela 5. Pliki modułu Visual Basic-deklaracje wywołań, typy danych, kody powrotu, stałe i struktury					
Nazwa pliku	**Opis**	**IBM i**	**Systemy AIX and Linux**	**Windows**	**z/OS**
Deklaracje wywołań, typy danych, kody powrotu, stałe i struktury					
CMQB	Definicje MQI			B	
KMQBB	Definicje MQAI			B	
CMQCFB	Definicje PCF			B	
CMQXB	Definicje kanału i wyjścia			B	
Klucz: B= Plik udostępniony					

z/OS *Pliki z/OS Assembler COPY*

Dostępne są różne pliki COPY, które ułatwiają pisanie aplikacji programu z/OS Assembler używających interfejsu MQI.

Tabela 6. Pliki kopii programu z/OS Assembler-typy danych, kody powrotu, stałe i struktury					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
Typy danych, kody powrotu i stałe					
KMQA	Definicje MQI				A
CMQCFA	Definicje PCF				A
CMQPSA	Definicje publikowania/subskrypcji				A
KMQVERA	Kontrola wersji struktury				A
Usługa CMQXA	Definicje kanału i wyjścia				A
Struktury					
CMQCDA	MQCD-definicja kanału				
CMQCFBFA	MQCFBF-parametr filtru łańcucha bajtów PCF				
CMQCFBSA	MQCFBS-parametr łańcucha bajtów PCF				A
CMQCFGRA	MQCFGR-parametr grupy PCF				A
CMQCFHA	MQCFH-nagłówek PCF				A
CMQCFIFA	MQCFIF-parametr filtru całkowitoliczbowego PCF				A
CMQCFILA	MQCFIL-parametr listy liczb całkowitych PCF				A
CMQCFINA	MQCFIN-parametr liczby całkowitej PCF				A
CMQCFSFA	MQCFSF-parametr filtru łańcucha PCF				A

Tabela 6. Pliki kopii programu z/OS Assembler-typy danych, kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQCFSLA	MQCFSL-parametr listy łańcuchów PCF				A
CMQCFSTA,	MQCFST-parametr łańcucha PCF				A
CMQCFXLA	MQCFIL64 -parametr listy 64-bitowych liczb całkowitych PCF				A
CMQCFXNA	MQCFIN64 -64-bitowy parametr całkowitoliczbowy PCF				A
KMQCHARVA	MQCHARV-łańcuch o zmiennej długości				A
CMQCIHA	MQCIH-nagłówek CICS bridge				A
CMQCNOA	MQCNO-opcje połączenia				A
CMQCSPA	MQCSP-parametry zabezpieczeń				A
CMQCXPA	MQCXP-parametry wyjścia kanału				A
CMQDHA	MQDH-nagłówek dystrybucji				A
CMQDLHA	MQDLH-nagłówek niedostarczonego komunikatu				A
CMQDXPA	MQDXP-parametry wyjścia konwersji danych				A
CMQEPHA	MQEPH-wbudowany nagłówek PCF				A
CMQGMOA	MQGMO-opcje pobierania komunikatu				A
CMQIIHA	MQIIH-nagłówek informacji IMS				A
CMQMDA	MQMD-deskryptor komunikatu				A
CMQMD1A	MQMD1 -deskryptor komunikatu w wersji 1				A
CMQMD2A	MQMD2 -deskryptor komunikatu w wersji 2				A
CMQMDEA	MQMDE-rozszerzony deskryptor komunikatu				A
CMQODA	MQOD-deskryptor obiektu				A
KMQORA	MQOR-rekord obiektu				A
CMQPMOA	MQPMO-opcje umieszczania komunikatów				A
CMQRFHA	MQRFH-reguły i nagłówek formatowania				A
CMQRFH2A	MQRFH2 -reguły i nagłówek formatowania 2				A
CMQRMHA	MQRMH-nagłówek komunikatu referencyjnego				A

Tabela 6. Pliki kopii programu z/OS Assembler-typy danych, kody powrotu, stałe i struktury (kontynuacja)					
Nazwa pliku	**Opis**	**IBM i**	**AIX and Linux**	**Windows**	**z/OS**
CMQTMA	MQTM-komunikat wyzwalacza				A
CMQTMC2A	MQTMC2 -komunikat wyzwalacza 2-znak				A
CMQWCRA.	MQWCR-Rekord klastra obciążenia klastra				A
CMQWDRA	MQWDR-Rekord docelowy obciążenia klastra				A
CMQWDR1A	MQWDR1 -wersja 1 rekordu docelowego obciążenia klastra				A
CMQWDR2A	MQWDR2 -wersja 2 rekordu docelowego obciążenia klastra				A
CMQWIHA	MQWIH-nagłówek informacji o pracy				A
CMQWQRA	MQWQR-Rekord kolejki obciążenia klastra				A
CMQWQR1A	MQWQR1 -Wersja 1 rekordu kolejki obciążenia klastra				A
CMQWQR2A	MQWQR2 -Rekord kolejki obciążenia klastra w wersji 2				A
CMQWXPA	MQWXP-Parametry wyjścia obciążenia klastra				A
CMQWXP1A	MQWXP1 -Parametry wyjścia obciążenia klastra w wersji 1				A
CMQWXP2A	MQWXP2 -Parametry wyjścia obciążenia klastra w wersji 2				A
CMQWXP3A	MQWXP3 -Parametry wyjścia obciążenia klastra w wersji 3				A
KMQXPA	MQXP-parametry wyjścia funkcji API języka CICS				A
CMQXQHA	MQXQH-nagłówek kolejki transmisji				A
CMQXWDA	MQXWD-Wyjście z deskryptora oczekiwania				A
Klucz: A= Plik udostępniony					

MQ_ * (Długości łańcuchów)

Tabela 7. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
DŁUGOŚĆ KODU MQ_ABEND_CODE_LENGTH	4	X'00000004'
MQ_ACCOUNTING_TOKEN_LENGTH,	32	X'00000020'
MQ_APPL_FUNCTION_NAME_LENGTH DŁUGOŚĆ	10	X'0000000A'

Tabela 7. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQ_APPL_IDENTITY_DATA_LENGTH,	32	X'00000020'
DŁUGOŚĆ_NAZWY_APLIKACJI_MQ	28	X'0000001C'
MQ_APPL_ORIGIN_DATA_LENGTH (długość danych aplikacji)	4	X'00000004'
DŁUGOŚĆ_TAGI_APLIKACJI_MQ	28	X'0000001C'
MQ_ARM_SUFFIX_LENGTH	2	X'00000002'
DŁUGOŚĆ CIĄGU MQ_ATTENTION_ID_LENGTH	4	X'00000004'
MQ_AUTH_INFO_CONN_NAME_LENGTH	264	X'00000108'
DŁUGOŚĆ PLIKU MQ_AUTH_INFO_DESC_LENGTH	64	X'00000040'
MQ_AUTH_INFO_XX_ENCODE_CASE_ONE długość_nazwy	48	X'00000030'
MQ_AUTH_INFO_OCSP_URL_LENGTH	256	X'00000100'
MQ_AUTHENTICATOR_LENGTH (długość elementu uwierzytelniającego MQ)	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 (długość nazwy mostu MQ)	24	X'00000018'
DŁUGOŚĆ KODU MQ_CANCEL_CODE_LENGTH	4	X'00000004'
MQ_CF_STRUC_DESC_LENGTH DŁUGOŚĆ	64	X'00000040'
MQ_CF_STRUC_NAME_LENGTH DŁUGOŚĆ	12	X'0000000C'
MQ_CHANNEL_DATE_LENGTH (DŁUGOŚĆ DANYCH KANAŁU)	12	X'0000000C'
MQ_CHANNEL_DESC_LENGTH	64	X'00000040'
MQ_CHANNEL_NAME_LENGTH (długość nazwy kanału)	20	X'00000014'
MQ_CHANNEL_TIME_LENGTH	8	X'00000008'
MQ_CHINIT_SERVICE_PARM_LENGTH,	32	X'00000020'
DŁUGOŚĆ NAZWY PLIKU MQ_CICS_FILE_NAME_LENGTH	8	X'00000008'
DŁUGOŚĆ IDENTYFIKATORA KLIENTA MQ_CLIENT_LENGTH	23	X'00000017'
MQ_CLUSTER_NAME_LENGTH (Długość nazwy klastra)	48	X'00000030'
DŁUGOŚĆ_NAZWY_KON_MQ	264	X'00000108'
MQ_CONN_TAG_LENGTH	128	X'00000080'
DŁUGOŚĆ IDENTYFIKATORA POŁĄCZENIA MQ_CONNECTION_LENGTH	24	X'00000018'
DŁUGOŚĆ IDENTYFIKATORA KORELACJI (MQ_CORREL_ID_LENGTH)	24	X'00000018'
MQ_CREATION_DATE_LENGTH (długość danych tworzenia)	12	X'0000000C'
MQ_CREATION_TIME_LENGTH (Długość czasu tworzenia)	8	X'00000008'
DŁUGOŚĆ_DANYCH_MQ	12	X'0000000C'
MQ_DISTINGUISHED_NAME_LENGTH,	1024	X'00000400'
DŁUGOŚĆ NAZWY GRUPY MQ_DNS_GROUP_NAME_LENGTH	18	X'00000012'
MQ_EXIT_DATA_LENGTH DŁUGOŚĆ	32	X'00000020'
DŁUGOŚĆ NAZWY MQ_EXIT_INFO_NAME_LENGTH	48	X'00000030'
DŁUGOŚĆ NAZWY WYJŚCIA MQ_EXIT_LENGTH	(value differs by platform or version)	

Tabela 7. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQ_EXIT_PD_AREA_LENGTH DŁUGOŚĆ	48	X'00000030'
MQ_EXIT_USER_AREA_LENGTH DŁUGOŚĆ	16	X'00000010'
MQ_FACILITY_LENGTH	8	X'00000008'
MQ_FACILITY_LIKE_LENGTH	4	X'00000004'
DŁUGOŚĆ_FORMATU_MQ	8	X'00000008'
MQ_FUNCTION_LENGTH,	4	X'00000004'
DŁUGOŚĆ GRUPY MQ_GROUP_ID_LENGTH	24	X'00000018'
DŁUGOŚĆ HASŁA MQ_LDAP_PASSWORD_LENGTH	32	X'00000020'
DŁUGOŚĆ_NAZWY_NASŁUCHIWANIA_MQ	48	X'00000030'
DŁUGOŚĆ OPISU MQ_LISTENER_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ_ADRESU_LOKALNEGO_MQ	48	X'00000030'
MQ_LTERM_OVERRIDE_LENGTH,	8	X'00000008'
DŁUGOŚĆ NAZWY MQ_LU_NAME_LENGTH	8	X'00000008'
DŁUGOŚĆ WŁAŚCIWOŚCI MQ_LUWID_LENGTH	16	X'00000010'
DŁUGOŚĆ NAZWY WYJŚCIA MQ_MAX_EXIT_LENGTH	128	X'00000080'
MQ_MAX_MCA_USER_ID_LENGTH	64	X'00000040'
DŁUGOŚĆ NAZWY WŁAŚCIWOŚCI MQ_MAX_PROPERTY_LENGTH	4095	X'00000FFF'
DŁUGOŚĆ IDENTYFIKATORA UŻYTKOWNIKA MQ_MAX_USER_ID_LENGTH	64	X'00000040'
DŁUGOŚĆ NAZWY ZADANIA MQ_MCA_JOB_LENGTH	28	X'0000001C'
DŁUGOŚĆ NAZWY MQ_MCA_LENGTH	20	X'00000014'
MQ_MCA_USER_DATA_LENGTH	32	X'00000020'
MQ_MCA_USER_ID_LENGTH	(value differs by platform or version)	(value differs by platform or version)
MQ_MFS_MAP_NAME_LENGTH,	8	X'00000008'
DŁUGOŚĆ NAZWY MODELU MQ_MODE_NAME_LENGTH	8	X'00000008'
MQ_MSG_HEADER_LENGTH (Długość sterty komunikatów)	4000	X'00000FA0'
DŁUGOŚĆ IDENTYFIKATORA KOMUNIKATU (MQ_MSG_ID_LENGTH)	24	X'00000018'
MQ_MSG_TOKEN_LENGTH DŁUGOŚĆ	16	X'00000010'
DŁUGOŚĆ OPISU LISTY MQ_NAMELIST_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ NAZWY LISTY MQ_NAMELIST_NAME_LENGTH	48	X'00000030'
MQ_NHA_INSTANCE_NAME_LENGTH DŁUGOŚĆ	48	X'00000030'
ID_INSTANCJI MQ_OBJECT_INSTANCE_LENGTH	24	X'00000018'
DŁUGOŚĆ NAZWY OBIEKTU MQ_OBJECT_NAME_LENGTH	48	X'00000030'
MQ_PASS_TICKET_APPL_LENGTH	8	X'00000008'
DŁUGOŚĆ HASŁA MQ_PASSWORD_LENGTH	12	X'0000000C'
DŁUGOŚĆ_ZASTOSOWANIA_PROCESU_MQ	256	X'00000100'
DŁUGOŚĆ PROCESU MQ_PROCESS_DESC_LENGTH	64	X'00000040'
MQ_PROCESS_ENV_DATA_LENGTH (długość danych ENV)	128	X'00000080'
DŁUGOŚĆ_NAZWY_PROCESU_MQ	48	X'00000030'

Tabela 7. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQ_PROCESS_USER_DATA_LENGTH,	128	X'00000080'
DŁUGOŚĆ NAZWY PROGRAMÓW MQ_PROGRAM_NAME_LENGTH	20	X'00000014'
DŁUGOŚĆ NAZWY APLIKACJI MQ_PUT_APPL_LENGTH	28	X'0000001C'
MQ_PUT_DATE_LENGTH DŁUGOŚĆ	8	X'00000008'
CZAS TRWANIA MQ_PUT_LENGTH	8	X'00000008'
MQ_Q_DESC_LENGTH DŁUGOŚĆ	64	X'00000040'
MQ_Q_MGR_DESC_LENGTH	64	X'00000040'
MQ_Q_MGR_IDENTIFIER_LENGTH,	48	X'00000030'
MQ_Q_MGR_NAME_LENGTH DŁUGOŚĆ	48	X'00000030'
MQ_Q_NAME_LENGTH (Długość nazwy kolejki)	48	X'00000030'
MQ_QSG_XX_ENCODE_CASE_ONE długość_nazwy	4	X'00000004'
MQ_REMOTE_SYS_ID_LENGTH DŁUGOŚĆ	4	X'00000004'
MQ_SECURITY_ID_LENGTH	40	X'00000028'
DŁUGOŚĆ WYBORU MQ_SELECTOR_LENGTH	10240	X'00002800'
DŁUGOŚĆ USŁUGI MQ_SERVICE_ARGS_LENGTH	255	X'000000FF'
MQ_SERVICE_COMMAND_LENGTH, DŁUGOŚĆ	255	X'000000FF'
DŁUGOŚĆ USŁUGI MQ_SERVICE_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ NAZWY USŁUGI (MQ_SERVICE_NAME_LENGTH)	32	X'00000020'
DŁUGOŚĆ ŚCIEŻKI USŁUGI MQ_SERVICE_PATH_LENGTH	255	X'000000FF'
MQ_SERVICE_STEP_LENGTH (Długość punktu końcowego usługi)	8	X'00000008'
MQ_SHORT_CONN_NAME_LENGTH DŁUGOŚĆ	20	X'00000014'
MQ_SHORT_DNAME_LENGTH DŁUGOŚĆ	256	X'00000100'
MQ_SSL_CIPHER_SPEC_LENGTH	32	X'00000020'
MQ_SSL_CRYPTO_HARDWARE_LENGTH,	256	X'00000100'
MQ_SSL_HANDSHAKE_STAGE_LENGTH DŁUGOŚĆ	32	X'00000020'
MQ_SSL_KEY_LIBRARY_LENGTH DŁUGOŚĆ	44	X'0000002C'
MQ_SSL_KEY_MEMBER_LENGTH,	8	X'00000008'
MQ_SSL_KEY_REPOSITORY_LENGTH (długość repozytorium kluczy SSL)	256	X'00000100'
MQ_SSL_PEER_NAME_LENGTH,	1024	X'00000400'
MQ_SSL_SHORT_PEER_NAME_LENGTH,	256	X'00000100'
DŁUGOŚĆ KODU MQ_START_CODE_LENGTH	4	X'00000004'
MQ_STORAGE_CLASS_DESC_LENGTH DŁUGOŚĆ	64	X'00000040'
MQ_STORAGE_CLASS_LENGTH DŁUGOŚĆ	8	X'00000008'
MQ_SUB_IDENTITY_LENGTH (DŁUGOŚĆ)	128	X'00000080'
MQ_SUB_POINT_LENGTH DŁUGOŚĆ	128	X'00000080'
MQ_SUITE_B_128_BIT	2	X'00000002'
MQ_SUITE_B_192_BIT	4	X'00000004'
MQ_SUITE_BNONE	1	X'00000001'
MQ_SUITE_BNOT_AVAILABLE	0	X'00000000'

Tabela 7. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
DŁUGOŚĆ NAZWY MQ_TCP_NAME_LENGTH	8	X'00000008'
DŁUGOŚĆ_CZASU_MQ	8	X'00000008'
DŁUGOŚĆ OPISU MQ_TOPIC_DESC_LENGTH	64	X'00000040'
DŁUGOŚĆ NAZWY MQ_TOPIC_NAME_LENGTH	48	X'00000030'
MQ_TOPIC_STR_LENGTH	10240	X'00002800'
MQ_TOTAL_EXIT_DATA_LENGTH	999	X'000003E7'
MQ_TOTAL_EXIT_NAME_LENGTH DŁUGOŚĆ	999	X'000003E7'
DŁUGOŚĆ NAZWY MQ_TP_NAME_LENGTH	64	X'00000040'
DŁUGOŚĆ NAZWY MQ_TPIPE_LENGTH	8	X'00000008'
MQ_TRAN_INSTANCE_ID_LENGTH DŁUGOŚĆ	16	X'00000010'
DŁUGOŚĆ IDENTYFIKATORA TRANSAKCJI (MQ_TRANSACTION_ID_LENGTH)	4	X'00000004'
MQ_TRIGGER_DATA_LENGTH (długość danych wyzwalacza MQ)	64	X'00000040'
MQ_TRIGGER_PROGRAM_NAME_LENGTH (długość programu wyzwalacza_mq_	8	X'00000008'
MQ_TRIGGER_TERM_ID_LENGTH DŁUGOŚĆ	4	X'00000004'
MQ_TRIGGER_TRANS_ID_LENGTH DŁUGOŚĆ	4	X'00000004'
ID_UŻYTKOWNIKA MQ_USER_ID_LENGTH	12	X'0000000C'
MQ_VERSION_LENGTH (długość wersji)	8	X'00000008'
MQ_XCF_GROUP_NAME_LENGTH,	8	X'00000008'
MQ_XCF_MEMBER_NAME_LENGTH,	16	X'00000010'

*MQ_ * (długość łańcucha w formacie komendy)*

Tabela 8. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQ_ARCHIVE_PFX_LENGTH DŁUGOŚĆ	36	X'00000024'
MQ_ARCHIVE_UNIT_LENGTH DŁUGOŚĆ	8	X'00000008'
DŁUGOŚĆ_USŁUGI_MQ	4	X'00000004'
DŁUGOŚĆ NAZWY PLIKU PROFILU MQ_AUTH_PROFILE_NAME_LENGTH	48	X'00000030'
MQ_CF_LEID_LENGTH	12	X'0000000C'
MQ_COMMAND_MQSC_LENGTH DŁUGOŚĆ	32768	X'00008000'
MQ_DATA_SET_NAME_LENGTH DŁUGOŚĆ	44	X'0000002C'
MQ_DB2_NAME_LENGTH	4	X'00000004'
DŁUGOŚĆ NAZWY MQ_DSG_NAME_LENGTH	8	X'00000008'
DŁUGOŚĆ_NAZWY_USŁUGI_MQ	1024	X'00000400'
MQ_ENV_INFO_LENGTH	96	X'00000060'
MQ_IP_ADDRESS_LENGTH (Długość adresu IP MSMQ)	48	X'00000030'
MQ_LOG_CORREL_ID_LENGTH DŁUGOŚĆ	8	X'00000008'
MQ_LOG_EXTENT_NAME_LENGTH DŁUGOŚĆ	24	X'00000018'
DŁUGOŚĆ ŚCIEŻKI DZIENNIKA MQ_LOG_PATH_LENGTH	1024	X'00000400'

Tabela 8. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQ_LRSN_LENGTH DŁUGOŚĆ	12	X'0000000C'
DŁUGOŚĆ NAZWY MQ_ORIGIN_NAME_LENGTH	8	X'00000008'
DŁUGOŚĆ NAZWY MQ_PSB_NAME_LENGTH	8	X'00000008'
MQ_PST_ID_LENGTH (Długość identyfikatora MSMQ)	8	X'00000008'
MQ_Q_MGR_CPF_LENGTH,	4	X'00000004'
DŁUGOŚĆ IDENTYFIKATORA ODPOWIEDZI MQ_RESPONSE_ID_LENGTH	24	X'00000018'
DŁUGOŚĆ_USŁUGI_MQ	16	X'00000010'
MQ_SECURITY_PROFILE_LENGTH	40	X'00000028'
MQ_SERVICE_COMPONENT_LENGTH (DŁUGOŚĆ USŁUGI)	48	X'00000030'
DŁUGOŚĆ_NAZWY_POD_USŁUGI_MQ	10240	X'00002800'
DŁUGOŚĆ USŁUGI MQ_SYSP_SERVICE_LENGTH	32	X'00000020'
DŁUGOŚĆ NAZWY SYSTEMU MQ_SYSTEM_NAME_LENGTH	8	X'00000008'
MQ_TASK_NUMBER_LENGTH,	8	X'00000008'
MQ_TPIPE_PFX_LENGTH	4	X'00000004'
DŁUGOŚĆ_ID_USŁUGI_PRODUKTU_MQ	256	X'00000100'
MQ_USER_DATA_LENGTH (długość danych użytkownika)	10240	X'00002800'
DŁUGOŚĆ WOLUMINU MQ_VOLSER_LENGTH	6	X'00000006'

MQACH_ * (struktura nagłówka obszaru łańcucha wyjścia funkcji API)

Tabela 9. Struktury stałych	
Nazwa	**Struktura**
MQACH_STRUC_ID	"ACH¬"
MQACH_STRUC_ID_ARRAY	'A','C','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 10. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQACH_VERSION_1	1	X'00000001'
MQACH_CURRENT_VERSION	1	X'00000001'
MQACH_LENGTH_1	(value differs by platform or version)	(value differs by platform or version)
MQACH_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQACT_ * (Token rozliczania)

Tabela 11. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQACT_NONE	X'00...00' (32 wartości null)
MQACT_NONE_ARRAY	'\0','\0',... (32 wartości null)

MQACT_ * (Opcje działania formatu komendy)

Tabela 12. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQACT_FORCE_REMOVE	1	X'00000001'
MQACT_ADVANCE_LOG,	2	X'00000002'
MQACT_COLLECT_STATISTICS	3	X'00000003'
MQACT_PUBSUB,	4	X'00000004'

MQACTP_ * (działanie)

Tabela 13. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQACTP_NOWA	0	X'00000000'
MQACTP_FORWARD	1	X'00000001'
ODPOWIEDŹ MQACTP_REPLY	2	X'00000002'
RAPORT MQACTP_REPORT	3	X'00000003'

MQACTT_ * (typy znaczników rozliczania)

Tabela 14. Wartości stałych

Nazwa	Wartość szesnastkowa
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'
Użytkownik MQACTT_USER	X'19'

MQADOPT_ * (przyjmij nowe sprawdzenia MCA i adoptuj nowe typy MCA)

Dołącz sprawdzenia nowego MCA

Tabela 15. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Dołącz nowe typy MCA

Tabela 16. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (struktura rekordu informacji uwierzytelniającej)

Tabela 17. Struktury stałych	
Nazwa	**Struktura**
Identyfikator struktury MQAIR_STRUC_ID	"AIR¬"
MQAIR_STRUC_ID_ARRAY	'A','I','R','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 18. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAIR_VERSION_1	1	X'00000001'
MQAIR_VERSION_2	2	X'00000002'
MQAIR_CURRENT_VERSION	2	X'00000002'

MQAIT_ * (typ informacji uwierzytelniającej)

Tabela 19. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAIT_ALL	0	X'00000000'
MQAIT_CRL_LDAP	1	X'00000001'
MQAIT_OCSP	2	X'00000002'
MQAIT_IDPW_OS	3	X'00000003'
MQAIT_IDPW_LDAP	4	X'00000004'

MQAS_ * (Wartości stanu asynchronicznego w formacie komendy)

Tabela 20. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAS_BRAK	0	X'00000000'
ROZPOCZĘTE MQAS	1	X'00000001'
MQAS_START_WAIT (OCZEKIWANIE_NA_URUCHOMIENIE)	2	X'00000002'
MQAS_ZATRZYMANY	3	X'00000003'
MQAS_ZAWIESZONA	4	X'00000004'
MQAS_SUSPENDED_TEMPORARY	5	X'00000005'
MQAS_AKTYWNE	6	X'00000006'

Tabela 20. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAS_INAKTYWNE	7	X'00000007'

MQAT_ * (Typy aplikacji umieszczających)

Tabela 21. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAT_UNKNOWN	-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'
Maszyny wirtualne MQAT	12	X'0000000C'
MQAT_GUARDIAN (strażnik MQ)	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'
UŻYTKOWNIK_MQAT	25	X'00000019'
MQAT_BROKER	26	X'0000001A'
MQAT_QMGR_PUBLISH	26	X'0000001A'
MQAT_JAVA	28	X'0000001C'
MQAT_DQM	29	X'0000001D'
INICJATOR MQAT_CHANNEL_INITIATOR	30	X'0000001E'
MQAT_WLM	31	X'0000001F'

Tabela 21. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAT_BATCH	32	X'00000020'
MQAT_RRS_BATCH,	33	X'00000021'
MQAT_SIB	34	X'00000022'
MQAT_DEFAULT	(value differs by platform or version)	(value differs by platform or version)
MQAT_USER_FIRST	65536	X'00010000'
Tabela MQAT_USER_LAST	999999999	X'3B9AC9FF'

MQAUTH_ * (wartości uprawnień w formacie komendy)

Tabela 22. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAUTH_BRAK	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_CREATE	6	X'00000006'
MQAUTH_DELETE	7	X'00000007'
MQAUTH_DISPLAY	8	X'00000008'
MQAUTH_INPUT	9	X'00000009'
MQAUTH_INQUIRE	10	X'0000000A'
DANE WYJŚCIOWE MQAUTH_OUTPUT	11	X'0000000B'
MQAUTH_PASS_ALL_CONTEXT	12	X'0000000C'
MQAUTH_PASS_IDENTITY_CONTEXT,	13	X'0000000D'
MQAUTH_SET	14	X'0000000E'
MQAUTH_SET_ALL_CONTEXT	15	X'0000000F'
MQAUTH_SET_IDENTITY_CONTEXT,	16	X'00000010'
MQAUTH_CONTROL	17	X'00000011'
MQAUTH_CONTROL_EXTENDED,	18	X'00000012'
MQAUTH_PUBLISH	19	X'00000013'
MQAUTH_SUBSCRIBE,	20	X'00000014'
MQAUTH_RESUME,	21	X'00000015'
SYSTEM MQAUTH_SYSTEM	22	X'00000016'

MQAUTHOPT_ * (opcje uprawnień dla formatu komendy)

Tabela 23. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAUTHOPT_CUMULATIVE	256	X'00000100'
MQAUTHOPT_ENTITY_EXPLICIT	1	X'00000001'

Tabela 23. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (struktura kontekstu wyjścia funkcji API)

Tabela 24. Struktury stałych	
Nazwa	**Struktura**
MQAXC_STRUC_ID (Identyfikator struktury MQAXC)	"AXC¬"
MQAXC_STRUC_ID_ARRAY	'A','X','C','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 25. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAXC_VERSION_1	1	X'00000001'
MQAXC_CURRENT_VERSION	1	X'00000001'

MQAXP_ * (struktura parametru wyjścia funkcji API)

Tabela 26. Struktury stałych	
Nazwa	**Struktura**
MQAXP_STRUC_ID	"AXP¬"
MQAXP_STRUC_ID_ARRAY	'A','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 27. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQAXP_VERSION_1	1	X'00000001'
MQAXP_VERSION_2	2	X'00000002'
MQAXP_CURRENT_VERSION	2	X'00000002'

MQBA_ * (selektory atrybutów bajtowych)

Tabela 28. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBA_FIRST	6001	X'00001771'
MQBA_LAST	8000	X'00001F40'

MQBACF_ * (Typy parametrów bajtowych formatu komendy)

Tabela 29. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBACF_FIRST	7001	X'00001B59'

Tabela 29. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBACF_EVENT_ACCOUNTING_TOKEN,	7001	X'00001B59'
MQBACF_EVENT_SECURITY_ID	7002	X'00001B5A'
MQBACF_RESPONSE_SET	7003	X'00001B5B'
MQBACF_RESPONSE_ID (identyfikator odpowiedzi MQBACF)	7004	X'00001B5C'
Identyfikator MQBACF_EXTERNAL_UOW_ID	7005	X'00001B5D'
MQBACF_CONNECTION_ID (Identyfikator połączenia MQBACF)	7006	X'00001B5E'
MQBACF_GENERIC_CONNECTION_ID (Identyfikator połączenia MQBACF)	7007	X'00001B5F'
MQBACF_ORIGIN_UOW_ID (identyfikator UUID)	7008	X'00001B60'
MQBACF_Q_MGR_UOW_ID	7009	X'00001B61'
MQBACF_ACCOUNTING_TOKEN,	7010	X'00001B62'
MQBACF_CORREL_ID (Identyfikator relacji MQ)	7011	X'00001B63'
MQBACF_GROUP_ID (identyfikator grupy MQBACF)	7012	X'00001B64'
MQBACF_MSG_ID,	7013	X'00001B65'
MQBACF_CF_LEID	7014	X'00001B66'
MQBACF_DESTINATION_CORREL_ID (identyfikator korelacji)	7015	X'00001B67'
MQBACF_SUB_ID (Identyfikator MQBACF)	7016	X'00001B68'
MQBACF_LAST_UŻYWANE	7016	X'00001B68'

MQBL_ * (długość buforu dla łańcuchów mqAddi mqSet)

Tabela 30. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBL_NULL_PRZERWANE	-1	X'FFFFFFFF'

MQBMHO_ * (Opcje i struktura obsługi od buforu do komunikatu)

Struktura opcji uchwytu buforu do komunikatu

Tabela 31. Struktury stałych	
Nazwa	**Struktura**
MQBMHO_STRUC_ID	"BMHO"
MQBMHO_STRUC_ID_ARRAY	'B','M','H','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 32. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBMHO_VERSION_1	1	X'00000001'
MQBMHO_CURRENT_VERSION	1	X'00000001'

Opcje uchwytu buforu do komunikatu

Tabela 33. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBMHO_NONE	0	X'00000000'
MQBMHO_DELETE_PROPERTIES	1	X'00000001'

MQBND_ * (Powiązania domyślne)

Tabela 34. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBND_BIND_ON_OPEN	0	X'00000000'
MQBND_BIND_NOT_FIXED (NIEUSTALONY)	1	X'00000001'
MQBND_BIND_ON_GROUP	2	X'00000002'

MQBO_ * (opcje i struktura początku)

Struktura opcji początku

Tabela 35. Struktury stałych	
Nazwa	**Struktura**
MQBO_STRUC_ID (identyfikator struktury MQBOC)	"BO¬¬"
MQBO_STRUC_ID_ARRAY	'B','O','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 36. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBO_VERSION_1	1	X'00000001'
MQBO_CURRENT_VERSION	1	X'00000001'

Opcje początku

Tabela 37. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBO_BRAK	0	X'00000000'

MQBT_ * (typy mostów w formacie komendy)

Tabela 38. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQBT_OTMA	1	X'00000001'

MQCA_ * (Selektory atrybutów znakowych)

Tabela 39. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCA_ADMIN_TOPIC_NAME	2105	X'00000839'
MQCA_ALTERATION_DATE	2027	X'000007EB'

Tabela 39. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCA_ALTERATION_TIME	2028	X'000007EC'
MQCA_APPL_ID	2001	X'000007D1'
MQCA_AUTH_INFO_CONN_NAME	2053	X'00000805'
MQCA_AUTH_INFO_DESC	2046	X'000007FE'
MQCA_AUTH_INFO_NAME	2045	X'000007FD'
MQCA_AUTH_INFO_OCSP_URL	2109	X'0000083D'
MQCA_AUTO_REORG_CATALOG	2091	X'0000082B'
MQCA_AUTO_REORG_START_TIME	2090	X'0000082A'
MQCA_BACKOUT_REQ_Q_NAME	2019	X'000007E3'
MQCA_BASE_OBJECT_NAME	2002	X'000007D2'
MQCA_BASE_Q_NAME	2002	X'000007D2'
MQCA_BATCH_INTERFACE_ID	2068	X'00000814'
MQCA_CF_STRUC_DESC	2052	X'00000804'
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'

Tabela 39. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCA_LDAP_PASSWORD	2048	X'00000800'
MQCA_LDAP_USER_NAME	2047	X'000007FF'
MQCA_LU_GROUP_NAME	2072	X'00000818'
MQCA_LU_NAME	2073	X'00000819'
MQCA_LU62_ARM_SUFFIX	2074	X'0000081A'
MQCA_MODEL_DURABLE_Q	2096	X'00000830'
MQCA_MODEL_NON_DURABLE_Q	2097	X'00000831'
MQCA_MONITOR_Q_NAME	2066	X'00000812'
MQCA_NAMELIST_DESC	2009	X'000007D9'
MQCA_NAMELIST_NAME	2010	X'000007DA'
MQCA_NAMES	2020	X'000007E4'
MQCA_PARENT	2102	X'00000836'
MQCA_PASS_TICKET_APPL	2086	X'00000826'
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'

Tabela 39. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCA_SSL_KEY_REPOSITORY	2049	X'00000801'
MQCA_STORAGE_CLASS	2022	X'000007E6'
MQCA_STORAGE_CLASS_DESC	2042	X'000007FA'
MQCA_SYSTEM_LOG_Q_NAME	2065	X'00000811'
MQCA_TCP_NAME	2075	X'0000081B'
MQCA_TOPIC_DESC	2093	X'0000082D'
MQCA_TOPIC_NAME	2092	X'0000082C'
MQCA_TOPIC_STRING_FILTER	2108	X'0000083C'
MQCA_TOPIC_STRING	2094	X'0000082E'
MQCA_TPIPE_NAME	2085	X'00000825'
MQCA_TRIGGER_CHANNEL_NAME	2064	X'00000810'
MQCA_TRIGGER_DATA	2023	X'000007E7'
MQCA_TRIGGER_PROGRAM_NAME	2062	X'0000080E'
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_ * (Typy parametrów znakowych formatu komendy)

Tabela 40. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_NAZWALIST_NAME	3005	X'00000BBD'
MQCACF_TO_NAZWALIST_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'
NAZWY_PROCESów_MQCACF_PROCESS_NAMES	3012	X'00000BC4'
MQCACF_NAMELIST_NAMES	3013	X'00000BC5'

Tabela 40. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
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'
NAZWA_APLIKACJI_W_PAMIĘCI_PODRĘCZNYCH	3024	X'00000BD0'
MQCACF_USER_IDENTIFIER (identyfikator użytkownika pamięci podręcznej)	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_CORREL_ID (Identyfikator relacji MQ)	3033	X'00000BD9'
MQCACF_PUBLISH_TIMESTAMP	3034	X'00000BDA'
MQCACF_STRING_DATA (dane buforowania MQ)	3035	X'00000BDB'
MQCACF_SUPPORTED_STREAM_NAME,	3036	X'00000BDC'
MQCACF_REG_TOPIC	3037	X'00000BDD'
MQCACF_REG_TIME	3038	X'00000BDE'
MQCACF_REG_ID_UŻYTKOWNIKA	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'
ID_UŻYTKOWNIKA 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'

Tabela 40. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_REG_SUB_NAME	3053	X'00000BED'
MQCACF_SUBSCRIPTION_IDENTITY (jednostka identyfikatora subskrypcji)	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'
MQCACF_DATA_SET_NAME	3059	X'00000BF3'
MQCACF_UOW_START_DATE (data rozpoczęcia jednostki MQCACF)	3060	X'00000BF4'
MQCACF_UOW_START_TIME	3061	X'00000BF5'
MQCACF_UOW_LOG_START_DATE (data rozpoczęcia dziennika MQCACF_UOW_LOG_	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_ENTITY_NAME	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'
MQCACF_LOG_PATH,	3074	X'00000C02'
MQCACF_COMMAND_MQSC	3075	X'00000C03'
MQCACF_Q_MGR_CPF	3076	X'00000C04'
MQCACF_USAGE_LOG_RBA,	3078	X'00000C06'
MQCACF_USAGE_LOG_LRSN,	3079	X'00000C07'
MQCACF_COMMAND_SCOPE	3080	X'00000C08'
MQCACF_ASID	3081	X'00000C09'
MQCACF_PSB_NAME	3082	X'00000C0A'
MQCACF_PST_ID (Identyfikator tabeli MQCACF)	3083	X'00000C0B'
NUMER ZADANIA MQCACF_TASK_NUMBER	3084	X'00000C0C'
MQCACF_TRANSACTION_ID (Identyfikator transakcji MQ)	3085	X'00000C0D'
MQCACF_Q_MGR_UOW_ID	3086	X'00000C0E'
MQCACF_ORIGIN_NAME (nazwa tabeli MQCACF)	3088	X'00000C10'
MQCACF_ENV_INFO	3089	X'00000C11'
MQCACF_SECURITY_PROFILE	3090	X'00000C12'
MQCACF_XX_ENCODE_CASE_ONE data_konfiguracji	3091	X'00000C13'
MQCACF_XX_ENCODE_CASE_ONE czas_konfiguracji	3092	X'00000C14'

Tabela 40. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_DATA	3098	X'00000C1A'
MQCACF_BACKUP_TIME	3099	X'00000C1B'
MQCACF_SYSTEM_NAME (nazwa systemu pamięci podręcznej MQ)	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'
KLASA MQCACF_FROM_STORAGE_CLASS	3104	X'00000C20'
MQCACF_TO_STORAGE_CLASS	3105	X'00000C21'
MQCACF_STORAGE_CLASS_NAMES,	3106	X'00000C22'
Nazwa_zestawu_pamięci_podręcznych MQF	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 (identyfikator korelacji dziennika MQCACF)	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 (nazwa nasłuchiwania MQCACF_FROM)	3124	X'00000C34'
MQCACF_TO_LISTENER_NAME (nazwa nasłuchiwania pamięci podręcznej MQ)	3125	X'00000C35'
MQCACF_FROM_NAZWA_USŁUGI	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 (data pobrania)	3130	X'00000C3A'
MQCACF_LAST_GET_TIME	3131	X'00000C3B'

Tabela 40. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_OPERATION_DATE (data operacji pamięci podręcznej MQ)	3132	X'00000C3C'
MQCACF_OPERATION_TIME (czas operacji pamięci podręcznej MQ)	3133	X'00000C3D'
MQCACF_ACTIVITY_DESC	3134	X'00000C3E'
MQCACF_APPL_IDENTITY_DATA	3135	X'00000C3F'
MQCACF_APPL_ORIGIN_DATA (DANE ZASTOSOWANIA)	3136	X'00000C40'
MQCACF_PUT_DATE (data)	3137	X'00000C41'
MQCACF_PUT_TIME	3138	X'00000C42'
MQCACF_REPLY_TO_Q	3139	X'00000C43'
MQCACF_REPLY_TO_Q_MGR	3140	X'00000C44'
MQCACF_RESOLVED_Q_NAME (nazwa kolejki pamięci podręcznej)	3141	X'00000C45'
MQCACF_STRUC_ID (identyfikator struktury pamięci podręcznej MQ)	3142	X'00000C46'
MQCACF_WARTOŚĆ_NAZWA	3143	X'00000C47'
MQCACF_SERVICE_START_DATE (data rozpoczęcia usługi)	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 (nazwa podrzędna pamięci podręcznej)	3152	X'00000C50'
MQCACF_DESTINATION_Q_MGR	3153	X'00000C51'
MQCACF_DESTINATION	3154	X'00000C52'
ID_UŻYTKOWNIKA MQCACF_SUB_USER_ID	3156	X'00000C54'
MQCACF_SUB_USER_DATA	3159	X'00000C57'
PODSELEKTOR MQCACF_SUB_SELECTOR	3160	X'00000C58'
MQCACF_LAST_PUB_DATE (data ostatniej aktualizacji pamięci podręcznej)	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'
MQCACF_FILTR	3170	X'00000C62'
MQCACF_BRAK	3171	X'00000C63'
MQCACF_ADMIN_TOPIC_NAMES	3172	X'00000C64'
MQCACF_ROUTING_FINGER_PRINT	3173	X'00000C65'
MQCACF_APPL_DESC	3174	X'00000C66'

Tabela 40. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACF_Q_MGR_START_DATE	3175	X'00000C67'
MQCACF_Q_MGR_START_TIME	3176	X'00000C68'
MQCACF_FROM_COMM_INFO_NAME	3177	X'00000C69'
MQCACF_TO_COMM_INFO_NAME	3178	X'00000C6A'
MQCACF_CF_OFFLOAD_SIZE1	3179	X'00000C6B'
MQCACF_CF_OFFLOAD_SIZE2	3180	X'00000C6C'
MQCACF_CF_OFFLOAD_SIZE3	3181	X'00000C6D'
MQCACF_CF_SMDS_GENERIC_NAME	3182	X'00000C6E'
MQCACF_CF_SMDS	3183	X'00000C6F'
MQCACF_RECOVERY_DATE (data odtwarzania pamięci podręcznej MQ)	3184	X'00000C70'
MQCACF_RECOVERY_TIME,	3185	X'00000C71'
MQCACF_CF_SMDSCONN	3186	X'00000C72'
MQCACF_CF_STRUC_NAME (nazwa struktury CF)	3187	X'00000C73'
MQCACF_ALTERNATE_ID_UŻYTKOWNIKA	3188	X'00000C74'
MQCACF_CHAR_ATTRS	3189	X'00000C75'
MQCACF_DYNAMIC_Q_NAME	3190	X'00000C76'
MQCACF_HOST_NAME	3191	X'00000C77'
MQCACF_MQCB_NAZWA	3192	X'00000C78'
MQCACF_OBJECT_STRING	3193	X'00000C79'
MQCACF_RESOLVED_LOCAL_Q_MGR	3194	X'00000C7A'
MQCACF_RESOLVED_LOCAL_Q_NAME,	3195	X'00000C7B'
MQCACF_RESOLVED_OBJECT_STRING	3196	X'00000C7C'
MQCACF_RESOLVED_Q_MGR	3197	X'00000C7D'
MQCACF_SELECTION_STRING (łańcuch MQCACF)	3198	X'00000C7E'
MQCACF_XA_INFO	3199	X'00000C7F'
MQCACF_APPL_FUNKCJA	3200	X'00000C80'
MQCACF_XQH_REMOTE_Q_NAME	3201	X'00000C81'
MQCACF_XQH_REMOTE_Q_MGR	3202	X'00000C82'
MQCACF_XQH_PUT_TIME	3203	X'00000C83'
MQCACF_XQH_PUT_DATE	3204	X'00000C84'
Komunikaty MQCACF_EXCL_OPERATOR_MESSAGES	3205	X'00000C85'
MQCACF_CSP_USER_IDENTIFIER	3206	X'00000C86'
MQCACF_AMQP_CLIENT_ID (Identyfikator klienta)	3207	X'00000C87'
MQCACF_ARCHIVE_LOG_EXTENT_NAME	3208	X'00000C88'
MQCACF_APPL_IMMOVABLE_DATE	3209	X'00000C89'
MQCACF_APPL_IMMOVABLE_TIME	3210	X'00000C8A'
MQCACF_NHA_INSTANCE_NAME,	3211	X'00000C8B'
MQCACF_LAST_UŻYWANE	3211	X'00000C8B'

MQCACH_ * (typy parametrów kanału znakowego formatu komendy)

Tabela 41. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCACH_FIRST	3501	X'00000DAD'
NAZWA KANAŁU 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'
Nazwa_połączenia_pamięci_podręcznej_MQ	3506	X'00000DB2'
MQCACH_MCA_NAME (nazwa pamięci podręcznej MQ)	3507	X'00000DB3'
MQCACH_SEC_EXIT_NAME	3508	X'00000DB4'
MQCACH_MSG_EXIT_NAME (nazwa wyjścia komunikatu pamięci podręcznej MQ)	3509	X'00000DB5'
MQCACH_SEND_EXIT_NAME (Nazwa wyjścia wysyłania pamięci podręcznej MQ)	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 (dane użytkownika wyjścia pamięci podręcznej MQ)	3516	X'00000DBC'
ID_UŻYTKOWNIKA MQCACH_USER_ID	3517	X'00000DBD'
HASŁO PAMIĘCI PODRĘCZNEJ MQCACHE	3518	X'00000DBE'
MQCACH_LOCAL_ADDRESS (Adres lokalny pamięci podręcznej MQ)	3520	X'00000DC0'
MQCACH_NAZWA_LOKALNA	3521	X'00000DC1'
MQCACH_LAST_MSG_TIME	3524	X'00000DC4'
MQCACH_LAST_MSG_DATE	3525	X'00000DC5'
MQCACH_MCA_ID_UŻYTKOWNIKA	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_NAZWA_WYJŚCIA	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'

Tabela 41. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCACH_REMOTE_APPL_TAG	3548	X'00000DDC'
MQCACH_SSL_CERT_ID_UŻYTKOWNIKA	3549	X'00000DDD'
MQCACH_SSL_CERT_ISSUER_NAME	3550	X'00000DDE'
MQCACH_LU_NAME	3551	X'00000DDF'
ADRES IP PAMIĘCI PODRĘCZNEJ MQCACHE	3552	X'00000DE0'
MQCACH_TCP_NAME	3553	X'00000DE1'
Nazwa obiektu nasłuchiwania MQCACH_LISTENER_NAME	3554	X'00000DE2'
MQCACH_LISTENER_DESC	3555	X'00000DE3'
MQCACH_LISTENER_START_DATE (data rozpoczęcia)	3556	X'00000DE4'
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_UŻYWANE	3559	X'00000DE7'

MQCADSD_ * (deskryptory ADS nagłówka informacyjnego CICS)

Tabela 42. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCADSD_BRAK	0	X'00000000'
MQCADSD_SEND	1	X'00000001'
MQCADSD_RECV	16	X'00000010'
MQCADSD_MSGFORMAT	256	X'00000100'

MQCAFTY_ * (wartości powinowactwa połączenia)

Tabela 43. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCAFTY_NONE	0	X'00000000'
MQCAFTY_PREFERRED	1	X'00000001'

MQCAMO_ * (typy parametrów monitorowania znaków w formacie komendy)

Tabela 44. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCAMO_FIRST	2701	X'00000A8D'
MQCAMO_CLOSE_DATE (data zamknięcia kolejki MQ)	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'

Tabela 44. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCAMO_OPEN_DATE (data otwarcia)	2709	X'00000A95'
MQCAMO_OPEN_TIME (czas otwarcia kolejki MQ)	2710	X'00000A96'
MQCAMO_START_DATE (data rozpoczęcia)	2711	X'00000A97'
MQCAMO_START_TIME	2712	X'00000A98'
MQCAMO_LAST_USED	2712	X'00000A98'

MQCBC_ * (struktura stałych MQCBC)

Tabela 45. Struktury stałych	
Nazwa	**Struktura**
MQCBC_ID_struktury	"CBC¬"
MQCBC_STRUC_ID_ARRAY	'C','B','C','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 46. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBC_VERSION_1	1	X'00000001'
MQCBC_CURRENT_VERSION	1	X'00000001'

MQCBCF_ * (flagi stałych MQCBC)

Tabela 47. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBCF_BRAK	0	X'00000000'
MQCBCF_READA_BUFFER_EMPTY	1	X'00000001'

MQCBCT_ * (typ wywołania zwrotnego stałych MQCBC)

Tabela 48. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBCT_START_CALL	1	X'00000001'
MQCBCT_STOP_CALL	2	X'00000002'
WYWOŁANIE MQCBCT_REGISTER_CALL	3	X'00000003'
WYWOŁANIA MQCBCT_DEREGISTER_CALL	4	X'00000004'
MQCBCT_EVENT_CALL	5	X'00000005'
MQCBCT_MSG_REMOVED	6	X'00000006'
MQCBCT_MSG_NOT_REMOVED (komunikat MQCBCT_NOT_REMOVED)	7	X'00000007'

MQCBD_ * (struktura stałych MQCBD)

Tabela 49. Struktury stałych	
Nazwa	**Struktura**
MQCBD_STRUC_ID (Identyfikator struktury CBC)	"CBD¬"

Tabela 49. Struktury stałych (kontynuacja)	
Nazwa	**Struktura**
MQCBD_STRUC_ID_ARRAY	'C','B','D','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 50. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBD_VERSION_1	1	X'00000001'
MQCBD_CURRENT_VERSION	1	X'00000001'

MQCBDO_ * (Opcje wywołania zwrotnego stałych MQCBD)

Tabela 51. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBDO_BRAK	0	X'00000000'
MQCBDO_START_CALL	1	X'00000001'
MQCBDO_STOP_CALL	4	X'00000004'
MQCBDO_REGISTER_CALL,	256	X'00000100'
WYWOŁANIA MQCBDO_DEREGISTER_CALL	512	X'00000200'
MQCBDO_FAIL_IF_QUIESCING,	8192	X'00002000'

MQCBO_ * (opcje tworzenia wielozbioru dla wielozbioru mqCreate)

Tabela 52. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBO_BRAK	0	X'00000000'
MQCBO_UŻYTKOWNIK_B	0	X'00000000'
MQCBO_ADMIN_BAG	1	X'00000001'
Komenda 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_INHIBITED	0	X'00000000'
MQCBO_REORDER_AS_REQUIRED	4	X'00000004'
MQCBO_DO_REORDER	0	X'00000000'
MQCBO_CHECK_SELECTORS	8	X'00000008'
MQCBO_DO_NOT_CHECK_SELECTORS	0	X'00000000'

MQCBT_ * (stałe MQCBD Jest to typ funkcji zwrotnej)

Tabela 53. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCBT_MESSAGE_CONSUMER	1	X'00000001'
MQCBT_EVENT_HANDLER	2	X'00000002'

MQCC_ * (kody zakończenia)

Tabela 54. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCC_OK	0	X'00000000'
Ostrzeżenie MQCC	1	X'00000001'
MQCC_FAILED (niepowodzenie MQC)	2	X'00000002'
MQCC_UNKNOWN	-1	X'FFFFFFFF'

MQCCSI_ * (identyfikatory kodowanego zestawu znaków)

Tabela 55. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0	X'00000000'
MQCCSI_DEFAULT	0	X'00000000'
MQCCSI_Q_MGR (menedżer kolejek MQ)	0	X'00000000'
MQCCSI_INHERIT	-2	X'FFFFFFFE'
MQCCSI_EMBEDDED	-1	X'FFFFFFFF'
APPL MQCCSI_PL	-3	X'FFFFFFFD'

MQCCT_ * (opcje zadania konwersacyjnego nagłówka informacji CICS)

Tabela 56. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCCT_YES	1	X'00000001'
MQCCT_NO	0	X'00000000'

MQCD_ * (struktura definicji kanału)

Tabela 57. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCD_VERSION_1	1	X'00000001'
MQCD_VERSION_2	2	X'00000002'
MQCD_VERSION_3	3	X'00000003'
MQCD_VERSION_4	4	X'00000004'
MQCD_VERSION_5	5	X'00000005'
MQCD_VERSION_6	6	X'00000006'
MQCD_VERSION_7	7	X'00000007'
MQCD_VERSION_8	8	X'00000008'
MQCD_VERSION_9	9	X'00000009'
MQCD_VERSION_10	10	X'0000000A'
Multi MQCD_VERSION_11	11	X'0000000B'
Multi BIEŻĄCA_WERSJA_MQCD	11	X'0000000B'
z/OS MQCD_VERSION_12	12	X'0000000C'

Tabela 57. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
z/OS BIEŻĄCA_WERSJA_MQCD	12	X'0000000C'
MQCD_LENGTH_4	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_5	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_6	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_7	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_8	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_9	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_10	(value differs by platform or version)	(value differs by platform or version)
MQCD_LENGTH_11	(value differs by platform or version)	(value differs by platform or version)
z/OS MQCD_LENGTH_12	(value differs by platform or version)	(value differs by platform or version)
MQCD_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQCDC_ * (Konwersja danych kanału)

Tabela 58. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
KONWERSJI MQCDC_SENDER_CONVERSION	1	X'00000001'
MQCDC_NO_SENDER_CONVERSION,	0	X'00000000'

MQCERT_ * (typ strategii sprawdzania poprawności certyfikatu)

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_ * (flagi możliwości)

Tabela 59. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCF_BRAK	0	X'00000000'
MQCF_DIST_LISTS	1	X'00000001'

MQCFAC_ * (narzędzie nagłówka informacji CICS)

Tabela 60. Nazwy i wartości stałych	
Nazwa	**Wartość szesnastkowa**
MQCFAC_BRAK	X'00...00' (8 wartości null)

Tabela 60. Nazwy i wartości stałych (kontynuacja)	
Nazwa	**Wartość szesnastkowa**
MQCFAC_NONE_ARRAY	`'\0','\0',...` (8 wartości null)

MQCFBF_ * (struktura parametru filtru łańcucha bajtów w formacie komendy)

Tabela 61. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFBF_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFBS_ * (struktura parametru łańcucha bajtowego formatu komendy)

Tabela 62. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFBS_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFC_ * (Opcje sterujące nagłówka formatu komendy)

Tabela 63. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFC_LAST	1	X'00000001'
MQCFC_NOT_LAST (nie)	0	X'00000000'

MQCFGR_ * (struktura parametru grupy formatu komendy)

Tabela 64. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFGR_STRUC_LENGTH DŁUGOŚĆ	16	X'00000010'

MQCFH_ * (struktura nagłówka formatu komendy)

Tabela 65. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (struktura parametru filtru liczb całkowitych w formacie komendy)

Tabela 66. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFIF_XX_ENCODE_CASE_ONE długość_struktury	20	X'00000014'

MQCFIL_ * (Struktura parametru listy liczb całkowitych w formacie komendy)

Tabela 67. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIL_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIL64_* (Struktura parametru 64-bitowej listy liczb całkowitych w formacie komendy)

Tabela 68. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIL64_STRUC_LENGTH_FIXED	16	X'00000010'

MQCFIN_ * (struktura parametru liczby całkowitej w formacie komendy)

Tabela 69. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
DŁUGOŚĆ STRUKTURY MQCFIN_STRUC_LENGTH	16	X'00000010'

MQCFIN64_* (struktura 64-bitowych parametrów całkowitoliczbowych w formacie komendy)

Tabela 70. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFIN64_STRUC_LENGTH	24	X'00000018'

MQCFO_ * (Opcje odświeżania repozytorium w formacie komendy i opcje usuwania kolejek w formacie komendy)

Opcje odświeżania repozytorium w formacie komend

Tabela 71. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFO_REFRESH_REPOSITORY_YES	1	X'00000001'
MQCFO_REFRESH_REPOSITORY_NO,	0	X'00000000'

Opcje usuwania kolejek w formacie komendy

Tabela 72. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFO_REMOVE_QUEUES_YES	1	X'00000001'
MQCFO_REMOVE_QUEUES_NO	0	X'00000000'

MQCFOP_ * (operatory filtru formatu komendy)

Tabela 73. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFOP_LESS	1	X'00000001'
MQCFOP_EQUAL	2	X'00000002'

Tabela 73. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFOP_GREATER	4	X'00000004'
MQCFOP_NOT_LESS,	6	X'00000006'
MQCFOP_NOT_EQUAL	5	X'00000005'
MQCFOP_NOT_GREATER (nie ZIELONY)	3	X'00000003'
MQCFOP_LIKE	18	X'00000012'
MQCFOP_NOT_LIKE (nie jest podobne do)	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_ * (CF Odtwarzalność)

Tabela 74. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFR_TAK	1	X'00000001'
MQCFR_NO	0	X'00000000'

MQCFSF_ * (struktura parametrów filtru łańcucha formatu komendy)

Tabela 75. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFSF_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFSL_ * (struktura parametru listy łańcuchów formatu komendy)

Tabela 76. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFSL_STRUC_LENGTH_FIXED	24	X'00000018'

MQCFST_ * (struktura parametrów formatu komendy)

Tabela 77. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFST_STRUC_LENGTH_FIXED	20	X'00000014'

MQCFSTATUS_ * (Status CF w formacie komendy)

Tabela 78. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFSTATUS_NOT_FOUND	0	X'00000000'
MQCFSTATUS_AKTYWNE	1	X'00000001'
MQCFSTATUS_IN_RECOVER	2	X'00000002'
MQCFSTATUS_W_KOPII zapasowej	3	X'00000003'
MQCFSTATUS_FAILED,	4	X'00000004'

Tabela 78. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFSTATUS_NONE	5	X'00000005'
MQCFSTATUS_NIEZNANY	6	X'00000006'
MQCFSTATUS_ADMIN_INCOMPLETE	20	X'00000014'
MQCFSTATUS_NEVER_USED	21	X'00000015'
MQCFSTATUS_NO_BACKUP	22	X'00000016'
MQCFSTATUS_NOT_FAILED (nie powiodło się)	23	X'00000017'
MQCFSTATUS_NOT_RECOVERABLE	24	X'00000018'
MQCFSTATUS_XES_BŁĄD	25	X'00000019'

MQCFT_ * (Typy struktur w formacie komend)

Tabela 79. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCFT_BRAK	0	X'00000000'
KOMENDA MQCFT_COMMAND	1	X'00000001'
ODPOWIEDZI_MQCFW	2	X'00000002'
MQCFT_INTEGER	3	X'00000003'
MQCFT_STRING	4	X'00000004'
MQCFT_INTEGER_LIST	5	X'00000005'
MQCFT_STRING_LIST (lista komend MQCF)	6	X'00000006'
ZDARZENIE_MQF	7	X'00000007'
MQCFT_USER	8	X'00000008'
MQCFT_BYTE_STRING	9	X'00000009'
MQCFT_TRACE_ROUTE,	10	X'0000000A'
RAPORT MQCF	12	X'0000000C'
FILTR MQCFT_INTEGER_FILTER	13	X'0000000D'
FILTR MQCFT_STRING_FILTER	14	X'0000000E'
MQCFT_BYTE_STRING_FILTR	15	X'0000000F'
MQCFT_COMMAND_XR	16	X'00000010'
MQCFT_XR_MSG	17	X'00000011'
MQCFT_XR_ELEMENT	18	X'00000012'
MQCFT_XR_SUMMARY	19	X'00000013'
GRUPA_MQCF	20	X'00000014'
MQCFT_XX_ENCODE_CASE_ONE statystyki	21	X'00000015'
MQCFT_ACCOUNTING (rozliczanie MQ)	22	X'00000016'
MQCFT_INTEGER64	23	X'00000017'
MQCFT_INTEGER64_LIST	25	X'00000019'

MQCFTYPE_ * (typy CF w formacie komendy)

Tabela 80. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCFTYPE_APPL	0	X'00000000'
MQCFTYPE_ADMIN	1	X'00000001'

MQCFUNC_ * (funkcje nagłówka informacji CICS)

Tabela 81. Struktury stałych

Nazwa	Struktura
MQCFUNC_MQCONN,	"CONN"
MQCFUNC_MQGET,	"GET¬"
MQCFUNC_MQINQ,	"INQ¬"
MQCFUNC_MQOPEN,	"OPEN"
MQCFUNC_MQPUT,	"PUT¬"
MQCFUNC_MQPUT1	"PUT1"
MQCFUNC_BRAK	"¬¬¬¬"
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	'¬','¬','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQCGWI_ * (CICS nagłówek informacji Get Wait Interval)

Tabela 82. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCGWI_DEFAULT	-2	X'FFFFFFFE'

MQCHAD_ * (automatyczna definicja kanału)

Tabela 83. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHAD_WYŁĄCZONE	0	X'00000000'
MQCHAD_ENABLED	1	X'00000001'

MQCHIDS_ * (Status wątpliwy w formacie komendy)

Tabela 84. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHIDS_NOT_INDOUBT.	0	X'00000000'
MQCHIDS_INDOUBT,	1	X'00000001'

MQCHLD_ * (Dyspozycje kanału w formacie komendy)

Tabela 85. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Komenda MQCHLD_ALL	-1	X'FFFFFFFF'
MQCHLD_DEFAULT	1	X'00000001'
MQCHLD_SHARED	2	X'00000002'
PRIVATE MQCHLD_PRIVATE	4	X'00000004'
MQCHLD_FIXSHARED	5	X'00000005'

MQCHS_ * (Status kanału w formacie komendy)

Tabela 86. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (Opcje współużytkowanego restartowania kanału formatu komendy)

Tabela 87. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHSH_RESTART_NO	0	X'00000000'
MQCHSH_RESTART_YES	1	X'00000001'

MQCHSR_ * (Opcje zatrzymania kanału dla formatu komendy)

Tabela 88. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHSR_STOP_NOT_REQUESTED,	0	X'00000000'
MQCHSR_STOP_REQUESTED	1	X'00000001'

MQCHSSTATE_ * (Podstany kanału w formacie komendy)

Tabela 89. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCHSSTATE_INNY	0	X'00000000'

Tabela 89. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCHSSTATE_END_OF_BATCH	100	X'00000064'
MQCHSSTATE_WYSYŁANIE	200	X'000000C8'
MQCHSSTATE_ODBIERANIE	300	X'0000012C'
MQCHSSTATE_SERIALIZING	400	X'00000190'
RESYNCHRONIZOWANIE MQCHSSTATE_RESYNCHING	500	X'000001F4'
MQCHSSTATE_HEARTPULSU	600	X'00000258'
MQCHSSTATE_IN_SCYEXIT	700	X'000002BC'
MQCHSSTATE_IN_RCVEXIT	800	X'00000320'
MQCHSSTATE_IN_SENDEXIT	900	X'00000384'
MQCHSSTATE_IN_MSGEXIT (MQCHSSTATE_MSGEXIT)	1000	X'000003E8'
MQCHSSTATE_IN_MREXIT (MQCHSSTATE_IN_REXIT)	1100	X'0000044C'
MQCHSSTATE_IN_CHADEXIT	1200	X'000004B0'
MQCHSSTATE_NET_CONNECTING	1250	X'000004E2'
MQCHSSTATE_SSL_HANDSHAKING,	1300	X'00000514'
SERWER_MQCHSSTATE_NAME_SERVER	1400	X'00000578'
MQCHSSTATE_IN_MQPUT	1500	X'000005DC'
MQCHSSTATE_IN_MQGET (MQCHSSTATE_MINIMUM)	1600	X'00000640'
MQCHSSTATE_IN_MQI_CALL,	1700	X'000006A4'
KOMPRESOWANIE MQCHSSTATE_COMPRESSING	1800	X'00000708'

MQCHT_ * (typy kanałów)

Tabela 90. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCHT_SENDER	1	X'00000001'
SERWER_MQ	2	X'00000002'
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_ * (typy tabel kanału formatu komendy)

Tabela 91. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCHTAB_Q_MGR	1	X'00000001'
MQCHTAB_CLNTCONN	2	X'00000002'

MQCI_ * (identyfikator korelacji)

Tabela 92. Nazwy i wartości stałych

Nazwa	Wartość
MQCI_BRAK	X'00...00' (24 wartości null)
MQCI_NONE_ARRAY	'\0','\0',... (24 wartości null)
MQCI_NEW_SESSION	X'414D5121...'
MQCI_NEW_SESSION_ARRAY	'\x41','\x4D','\51','\x21',...

MQCIH_ * (struktura i flagi nagłówka informacji CICS)

Struktura nagłówka informacji CICS

Tabela 93. Struktury stałych

Nazwa	Struktura
MQCIH_ID_struktury	"CIH¬"
MQCIH_STRUC_ID_ARRAY	'C','I','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 94. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

CICS flagi nagłówka informacji

Tabela 95. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCIH_BRAK	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_ * (typy pamięci podręcznej klastra)

Tabela 96. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCLCT_STATIC	0	X'00000000'
MQCLCT_DYNAMIC,	1	X'00000001'

MQCLRS_ * (Format komendy-Wyczyść zasięg łańcucha tematu)

Tabela 97. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCLRS_XX_ENCODE_CASE_ONE lokalne	1	X'00000001'
MQCLRS_GLOBAL	2	X'00000002'

MQCLRT_ * (Format komendy-Wyczyść typ łańcucha tematu)

Tabela 98. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCLRT_ZACHOWANY	1	X'00000001'

MQCLT_ * (typy odsyłaczy nagłówka informacji CICS)

Tabela 99. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCLT_PROGRAM	1	X'00000001'
MQCLT_TRANSACTION	2	X'00000002'

MQCLWL_ * (obciążenie klastra)

Tabela 100. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCLWL_USEQ_LOCAL.	0	X'00000000'
MQCLWL_USEQ_ANY	1	X'00000001'
MQCLWL_USEQ_AS_Q_MGR	-3	X'FFFFFFFD'

MQCLXQ_ * (typ kolejki transmisji klastra)

MQCLXQ_* to wartości, które można ustawić w atrybucie menedżera kolejek DEFCLXQ . Atrybut **DEFCLXQ** określa, która kolejka transmisji jest domyślnie wybierana przez kanały nadawcze klastra, z których mają być pobierane komunikaty, w celu wysyłania komunikatów do kanałów odbiorczych klastra.

Tabela 101. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCLXQ_SCTQ	0	X'00000000'
MQCLXQ_CHANNEL	1	X'00000001'

Odsyłacze pokrewne

“DefClusterXmitQueueTyp (MQLONG)” na stronie 843
Atrybut DefClusterXmitQueueType określa, która kolejka transmisji jest domyślnie wybierana przez kanały nadawcze klastra, z których mają być wysyłane komunikaty do kanałów odbiorczych klastra.

Zmiana menedżera kolejek

Sprawdź menedżera kolejek

Sprawdzanie menedżera kolejek (odpowiedź)

“MQINQ-zapytanie o obiekt-atrybuty” na stronie 727
Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierający atrybuty obiektu.

MQCMD_ * (kody komend)

Tabela 102. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_NONE	0	X'00000000'
MQCMD_CHANGE_Q_MGR,	1	X'00000001'
MQCMD_INQUIRE_Q_MGR,	2	X'00000002'
MQCMD_CHANGE_PROCESS,	3	X'00000003'
MQCMD_COPY_PROCESS	4	X'00000004'
MQCMD_CREATE_PROCESS	5	X'00000005'
PROCES MQCMD_DELETE_PROCESS	6	X'00000006'
MQCMD_INQUIRE_PROCESS,	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 (kolejka MQCM)	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'
MQCMD_COPY_CHANNEL (kanał kopii komend MQ)	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 (nasłuchiwanie kanału MQCM)	31	X'0000001F'
MQCMD_CHANGE_NAMELIST,	32	X'00000020'
MQCMD_COPY_NAMELIST (lista nazw kopii)	33	X'00000021'
MQCMD_CREATE_NAMELIST,	34	X'00000022'
MQCMD_DELETE_NAMELIST (lista nazw użytkownika)	35	X'00000023'
MQCMD_INQUIRE_NAMELIST (lista nazw zapytań mqcmd)	36	X'00000024'
MQCMD_INQUIRE_NAMELIST_NAMES	37	X'00000025'
MQCMD_ESCAPE	38	X'00000026'
MQCMD_RESOLVE_CHANNEL,	39	X'00000027'

Tabela 102. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_PING_Q_MGR (menedżer kolejek MQ)	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 (zdarzenie kanału MQCM)	46	X'0000002E'
PUBLIKACJA MQCMD_DELETE_PUBLACJI	60	X'0000003C'
MQCMD_DEREGISTER_PUBLISHER	61	X'0000003D'
MQCMD_DEREGISTER_SUBSCRIBER	62	X'0000003E'
MQCMD_PUBLISH	63	X'0000003F'
MQCMD_REGISTER_PUBLISHER	64	X'00000040'
MQCMD_REGISTER_SUBSCRIBER	65	X'00000041'
MQCMD_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'
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'
ZDARZENIE MQCMD_LOGGER_EVENT	91	X'0000005B'
MQCMD_RESET_Q_MGR	92	X'0000005C'
MQCMD_CHANGE_LISTENER,	93	X'0000005D'
MQCMD_COPY_LISTENER (proces nasłuchujący kopii MQ)	94	X'0000005E'

Tabela 102. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_CREATE_LISTENER	95	X'0000005F'
MQCMD_DELETE_LISTENER,	96	X'00000060'
MQCMD_INQUIRE_LISTENER,	97	X'00000061'
STATUS LISTENER_MQCMD_INQUIRE_LISTENER_STATUS	98	X'00000062'
MQCMD_COMMAND_EVENT (zdarzenie MQCM)	99	X'00000063'
MQCMD_CHANGE_SECURITY	100	X'00000064'
MQCMD_CHANGE_CF_STRUC	101	X'00000065'
KLASA MQCMD_CHANGE_STG_CLASS	102	X'00000066'
MQCMD_CHANGE_TRACE (śledzenie zmian MQCM)	103	X'00000067'
DZIENNIK_ARCHIWUM_MQCM	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 (klasa STG)	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'
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'
ŚLEDZENIE MQCMD_INQUIRE_TRACE	125	X'0000007D'
MQCMD_INQUIRE_USAGE	126	X'0000007E'
MQCMD_MOVE_Q (kolejka MQCM)	127	X'0000007F'
MQCMD_RECOVER_BSDS	128	X'00000080'
MQCMD_RECOVER_CF_STRUC	129	X'00000081'
MQCMD_RESET_TPIPE	130	X'00000082'
MQCMD_RESOLVE_INDOUBT	131	X'00000083'
MQCMD_RESUME_Q_MGR	132	X'00000084'
MQCMD_REVERIFY_SECURITY	133	X'00000085'

Tabela 102. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCMD_SET_ARCHIVE,	134	X'00000086'
MQCMD_SET_LOG,	136	X'00000088'
MQCMD_SET_SYSTEM	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'
USŁUGA_ZMIANY_MQCM	149	X'00000095'
Usługa MQCMD_COPY_SERVICE	150	X'00000096'
MQCMD_CREATE_SERVICE (usługa MQCM)	151	X'00000097'
Usługa MQCMD_DELETE_SERVICE	152	X'00000098'
MQCMD_INQUIRE_SERVICE	153	X'00000099'
MQCMD_INQUIRE_SERVICE_STATUS	154	X'0000009A'
Usługa MQCMD_START_SERVICE	155	X'0000009B'
MQCMD_STOP_SERVICE	156	X'0000009C'
MQCMD_DELETE_BUFFER_POOL	157	X'0000009D'
ZESTAW 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'
Liczba operacji MQCMD_ACCOUNTING_MQI	167	X'000000A7'
LICZNIK MQCMD_ACTING_Q	168	X'000000A8'
MQCMD_INQUIRE_AUTH_SERVICE	169	X'000000A9'
MQCMD_CHANGE_TOPIC	170	X'000000AA'
MQCMD_COPY_TOPIC	171	X'000000AB'
MQCMD_CREATE_TOPIC	172	X'000000AC'
MQCMD_DELETE_TOPIC,	173	X'000000AD'
MQCMD_INQUIRE_TOPIC (temat zapytania MQ)	174	X'000000AE'

Tabela 102. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'
KANAŁ_CZYSZCZENIA_MQCM	195	X'000000C3'

MQCMDI_ * (Wartości informacji o komendzie w formacie komendy)

Tabela 103. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCMDI_CMDSCOPE_ACCEPTED	1	X'00000001'
MQCMDI_CMDSCOPE_GENERATED	2	X'00000002'
MQCMDI_CMDSCOPE_COMPLETED	3	X'00000003'
MQCMDI_QSG_DISP_COMPLETED,	4	X'00000004'
MQCMDI_COMMAND_ACCEPTED	5	X'00000005'
MQCMDI_CLUSTER_REQUEST_W_KOLEJCE	6	X'00000006'
MQCMDI_CHANNEL_INIT_STARTED	7	X'00000007'
MQCMDI_RECOVER_STARTED,	11	X'0000000B'
MQCMDI_BACKUP_URUCHOMIONY	12	X'0000000C'
MQCMDI_RECOVER_COMPLETED	13	X'0000000D'
MQCMDI_SEC_TIMER_ZERO	14	X'0000000E'
MQCMDI_REFRESH_CONFIGURATION	16	X'00000010'
MQCMDI_SEC_SIGNOFF_ERROR (BŁĄD PODPISU MQCMDI)	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_WIELKIE litery	21	X'00000015'
MQCMDI_SEC_MIXEDCASE	22	X'00000016'

MQCMDL_ * (poziomy komend)

Tabela 104. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQCMDL_LEVEL_800	800
MQCMDL_LEVEL_801	801
MQCMDL_LEVEL_802	802

Tabela 104. Nazwy i wartości stałych (kontynuacja)	
Nazwa	**Wartość**
MQCMDL_LEVEL_900	900
MQCMDL_LEVEL_901	901
MQCMDL_LEVEL_902	902
MQCMDL_LEVEL_903	903
MQCMDL_LEVEL_904	904
MQCMDL_LEVEL_905	905
MQCMDL_LEVEL_910	910
MQCMDL_LEVEL_912	912
MQCMDL_LEVEL_913	913
MQCMDL_LEVEL_914	914
MQCMDL_LEVEL_915	915
MQCMDL_LEVEL_920	920
MQCMDL_LEVEL_921	921
MQCMDL_LEVEL_922	922
MQCMDL_LEVEL_923	923
MQCMDL_LEVEL_924	924
MQCMDL_LEVEL_925	925
MQCMDL_LEVEL_930	930
MQCMDL_LEVEL_931	931
MQCMDL_LEVEL_932	932

MQCMHO_ * (Tworzenie opcji i struktury uchwytu komunikatu)

Utwórz strukturę opcji uchwytu komunikatu

Tabela 105. Struktury stałych	
Nazwa	**Struktura**
MQCMHO_STRUC_ID (Identyfikator struktury MQCM)	"CMHO"
MQCMHO_STRUC_ID_ARRAY	'C','M','H','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 106. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCMHO_VERSION_1	1	X'00000001'
MQCMHO_CURRENT_VERSION	1	X'00000001'

Opcje tworzenia uchwytu komunikatu

Tabela 107. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCMHO_DEFAULT_VALIDATION,	0	X'00000000'
MQCMHO_NO_VALIDATION,	1	X'00000001'

Tabela 107. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCMHO_VALIDATE,	2	X'00000002'
MQCMHO_NONE	0	X'00000000'

MQCNO_ * (opcje i struktura połączenia)

Struktura opcji połączenia

Tabela 108. Struktury stałych	
Nazwa	**Struktura**
MQCNO_STRUC_ID	"CNO¬"
MQCNO_STRUC_ID_ARRAY	'C','N','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 109. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'
MQCNO_CURRENT_VERSION	5	X'00000005'

Opcje połączenia

Tabela 110. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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 (mqcno_restrict_conn_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_IZOLOWANE_POWIĄZANIE	512	X'00000200'
POWIĄZANIE_LOKALNE_MQCNO_BINDING	1024	X'00000400'
MQCNO_CLIENT_BINDING	2048	X'00000800'
MQCNO_ACCOUNTING_MQI_ENABLED,	4096	X'00001000'
MQCNO_ACCOUNTING_MQI_WYŁĄCZONE	8192	X'00002000'

Tabela 110. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCNO_ACCOUNTING_Q_ENABLED,	16384	X'00004000'
MQCNO_ACCOUNTING_Q_DISABLED	32768	X'00008000'
MQCNO_NO_CONV_SHARING,	65536	X'00010000'
MQCNO_ALL_CONVS_SHARE	262144	X'00040000'
MQCNO_CD_FOR_OUTPUT_ONLY,	524288	X'00080000'
MQCNO_USE_CD_SELECTION	1048576	X'00100000'
MQCNO_RECONNECT	16777216	X'01000000'
MQCNO_RECONNECT_AS_DEF	0	X'00000000'
MQCNO_RECONNECT_DISABLED	33554432	X'02000000'
MQCNO_RECONNECT_Q_MGR	67108864	X'04000000'
MQCNO_ACTIVITY_TRACE_ENABLED,	134217728	X'08000000'
MQCNO_ACTIVITY_TRACE_DISABLED	268435456	X'10000000'
MQCNO_BRAK	0	X'00000000'

MQCO_ * (Opcje zamykania)

Tabela 111. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCO_IMMEDIATE	0	X'00000000'
MQCO_BRAK	0	X'00000000'
MQCO_DELETE	1	X'00000001'
MQCO_DELETE_PURGE	2	X'00000002'
MQCO_KEEP_SUB	4	X'00000004'
MQCO_REMOVE_SUB	8	X'00000008'
MQCO_QUIESCE,	32	X'00000020'

MQCODL_ * (CICS długość danych wyjściowych nagłówka informacji)

Tabela 112. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCODL_AS_INPUT	-1	X'FFFFFFFF'

MQCOMPRESS_ * (Kompresja kanału)

Tabela 113. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCOMPRESS_NOT_AVAILABLE	-1	X'FFFFFFFF'
MQCOMPRESS_NONE	0	X'00000000'
MQCOMPRESS_RLE	1	X'00000001'
MQCOMPRESS_ZLIBFAST,	2	X'00000002'
MQCOMPRESS_ZLIBHIGH	4	X'00000004'
SYSTEM MQCOMPRESS_SYSTEM	8	X'00000008'
MQCOMPRESS_ANY	268435455	X'0FFFFFFF'

MQCONNID_ * (identyfikator połączenia)

Tabela 114. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQCONNID_NONE	X'00...00' (24 wartości null)
MQCONNID_NONE_ARRAY	'\0','\0',... (24 wartości null)

MQCOPY_ * (opcje kopiowania właściwości)

Tabela 115. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCOPY_NONE	0	X'00000000'
MQCOPY_ALL	1	X'00000001'
MQCOPY_FORWARD	2	X'00000002'
MQCOPY_PUBLISH	4	X'00000004'
MQCOPY_REPLY	8	X'00000008'
MQCOPY_REPORT	16	X'00000010'
MQCOPY_DEFAULT	22	X'00000016'

MQCQT_ * (typy kolejek klastra)

Tabela 116. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (CICS -nagłówek informacji-kody powrotu)

Tabela 117. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQCRC_OK	0	X'00000000'
BŁĄD WYKONYWANIA MQCRC_CICS_EXEC_ERROR	1	X'00000001'
BŁĄD MQCRC_MQ_API_ERROR	2	X'00000002'
BŁĄD MQCRC_BRIDGE_ERROR	3	X'00000003'
MQCRC_BRIDGE_ABEND, nieprawidłowe zakończenie	4	X'00000004'
MQCRC_APPLICATION_ABEND (niepoprawne zakończenie aplikacji MQ)	5	X'00000005'
BŁĄD MQCRC_SECURITY_ERROR	6	X'00000006'
MQCRC_PROGRAM_NIEDOSTĘPNE	7	X'00000007'
MQCRC_BRIDGE_TIMEOUT,	8	X'00000008'
MQCRC_TRANSID_NIEDOSTĘPNE	9	X'00000009'

MQCS_ * (stałe MQCBC, stan konsumenta)

Tabela 118. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCS_NONE	0	X'00000000'
MQCS_SUSPENDED_TEMPORARY	1	X'00000001'
MQCS_SUSPENDED_USER_ACTION	2	X'00000002'
MQCS_SUSPENDED	3	X'00000003'
MQCS_ZATRZYMANY	4	X'00000004'

MQCSC_ * (CICS nagłówek informacji o kodach początkowych)

Tabela 119. Struktury stałych

Nazwa	Struktura
MQCSC_START	"S¬¬¬"
MQCSC_STARTDATA	"SD¬¬"
MQCSC_TERMINPUT	"TD¬¬"
MQCSC_BRAK	"¬¬¬¬"
MQCSC_START_ARRAY	'S','¬','¬','¬'
MQCSC_STARTDATA_ARRAY	'S','D','¬','¬'
MQCSC_TERMINPUT_ARRAY	'T','D','¬','¬'
MQCSC_NONE_ARRAY	'¬','¬','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQCSP_ * (struktura parametrów zabezpieczeń połączenia i typy uwierzytelniania)

Struktura parametrów zabezpieczeń połączenia

Tabela 120. Struktury stałych

Nazwa	Struktura
MQCSP_STRUC_ID	"CSP¬"
MQCSP_STRUC_ID_ARRAY	'C','S','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 121. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCSP_VERSION_1	1	X'00000001'
MQCSP_VERSION_2	2	X'00000002'
MQCSP_VERSION_3	3	X'00000003'
MQCSP_CURRENT_VERSION	3	X'00000003'

Typy uwierzytelniania parametrów zabezpieczeń połączenia

Tabela 122. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCSP_AUTH_BRAK	0	X'00000000'
MQCSP_AUTH_ID_UŻYTKOWNIKA AND_PWD	1	X'00000001'
MQCSP_AUTH_ID_TOKEN	2	X'00000002'

MQCSRV_ * (opcje serwera komend)

Tabela 123. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (znacznik połączenia menedżera kolejek)

Tabela 124. Nazwy i wartości stałych

Nazwa	Wartość
MQCT_NONE	X'00...00' (128 wartości null)
MQCT_NONE_ARRAY	'\0','\0',... (128 wartości null)

MQCTES_ * (CICS informacyjny nagłówek Status zakończenia zadania)

Tabela 125. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCTES_NOSYNC	0	X'00000000'
MQCTES_COMMIT	256	X'00000100'
MQCTES_BACKOUT	4352	X'00001100'
MQCTES_ENDTASK	65536	X'00010000'

MQCTLO_ * (struktura opcji MQCTL i opcje kontroli konsumenta)

Struktura opcji MQCTL

Tabela 126. Struktury stałych

Nazwa	Struktura
Identyfikator struktury MQCTLO_STRUC_ID	"CTLO"
MQCTLO_STRUC_ID_ARRAY	'C','T','L','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 127. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCTLO_VERSION_1	1	X'00000001'
MQCTLO_CURRENT_VERSION	1	X'00000001'

Opcje MQCTL Opcje kontroli konsumenta

Tabela 128. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCTLO_NONE	0	X'00000000'
MQCTLO_THREAD_AFFINITY	1	X'00000001'
MQCTLO_FAIL_IF_QUIESCING,	8192	X'00002000'

MQCUOWC_ * (elementy sterujące jednostki pracy w nagłówku informacji CICS)

Tabela 129. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
TYLKO 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_ * (struktura parametru wyjścia kanału)

Tabela 130. Struktury stałych

Nazwa	Struktura
MQCXP_STRUC_ID	"CXP¬"
MQCXP_STRUC_ID_ARRAY	'C','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 131. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQCXP_VERSION_1	1	X'00000001'
MQCXP_VERSION_2	2	X'00000002'
MQCXP_VERSION_3	3	X'00000003'
MQCXP_VERSION_4	4	X'00000004'
MQCXP_VERSION_5	5	X'00000005'
MQCXP_VERSION_6	6	X'00000006'
MQCXP_VERSION_7	7	X'00000007'
MQCXP_VERSION_8	8	X'00000008'
MQCXP_VERSION_9	9	X'00000009'
MQCXP_CURRENT_VERSION	9	X'00000009'

MQDC_ * (Klasa docelowa)

Tabela 132. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDC_MANAGED (zarządzany przez MQDC)	1	X'00000001'
DOSTARCZONA MQDC	2	X'00000002'

MQDCC_ * (opcje konwersji, maski i czynniki)

Opcje konwersji

Tabela 133. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDCC_DEFAULT_CONVERSION	1	X'00000001'
MQDCC_FILL_TARGET_BUFFER	2	X'00000002'
MQDCC_INT_DEFAULT_XX_ENCODE_CASE_ONE konwersja	4	X'00000004'
MQDCC_SOURCE_ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_SOURCE_ENC_NORMAL	16	X'00000010'
MQDCC_SOURCE_ENC_REVERSED	32	X'00000020'
MQDCC_SOURCE_ENC_NIEZDEFINIOWANY	0	X'00000000'
MQDCC_TARGET__ENC_NATIVE	(value differs by platform or version)	(value differs by platform or version)
MQDCC_TARGET_ENC_NORMAL	256	X'00000100'
MQDCC_TARGET_ENC_REVERSED	512	X'00000200'
MQDCC_TARGET_ENC_UNDEFINED	0	X'00000000'
MQDCC_BRAK	0	X'00000000'

Maski i czynniki opcji konwersji

Tabela 134. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDCC_SOURCE_ENC_MASK,	240	X'000000F0'
MQDCC_TARGET_ENC_MASK	3840	X'00000F00'
MQDCC_SOURCE_ENC_FACTOR	16	X'00000010'
MQDCC_TARGET_FACTOR	256	X'00000100'

MQDELO_ * (opcje usuwania publikowania/subskrypcji)

Tabela 135. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDELO_NONE	0	X'00000000'
MQDELO_LOCAL (LOKALNE)	4	X'00000004'

MQDH_ * (struktura nagłówka dystrybucji)

Tabela 136. Struktury stałych	
Nazwa	**Struktura**
ID struktury MQDH_ID	"DH¬¬"
MQDH_STRUC_ID_ARRAY	'D','H','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 137. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDH_VERSION_1	1	X'00000001'
MQDH_CURRENT_VERSION	1	X'00000001'

MQDHF_ * (flagi nagłówka dystrybucji)

Tabela 138. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDHF_NEW_MSG_IDS	1	X'00000001'
MQDHF_BRAK	0	X'00000000'

MQDISCONNECT_ * (Typy rozłączania formatu komendy)

Tabela 139. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDISCONNECT_NORMAL	0	X'00000000'
MQDISCONNECT_IMPLICIT,	1	X'00000001'
MQDISCONNECT_Q_MGR	2	X'00000002'

MQDL_ * (listy dystrybucyjne)

Tabela 140. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDL_SUPPORTED (obsługiwane)	1	X'00000001'
NIEOBSŁUGIWANA MQDL_NOT_SUPPORTED	0	X'00000000'

MQDLH_ * (struktura nagłówka niedostarczonego komunikatu)

Tabela 141. Struktury stałych	
Nazwa	**Struktura**
MQDLH_ID_struktury	"DLH¬"
MQDLH_STRUC_ID_ARRAY	'D','L','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQDLH_VERSION_1	1	X'00000001'
MQDLH_CURRENT_VERSION	1	X'00000001'

MQDLV_ * (Trwałe/Nietrwałe dostarczanie komunikatów)

Tabela 142. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDLV_AS_PARENT	0	X'00000000'
MQDLV_WSZYSTKIE	1	X'00000001'
MQDLV_ALL_DUR	2	X'00000002'
MQDLV_ALL_AVAIL	3	X'00000003'

MQDMHO_ * (Usunięcie opcji i struktury uchwytu komunikatu)

Usuń strukturę opcji uchwytu komunikatu

Tabela 143. Struktury stałych

Nazwa	Struktura
MQDMHO_STRUC_ID	"DMHO"
MQDMHO_STRUC_ID_ARRAY	'D','M','H','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 144. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDMHO_VERSION_1	1	X'00000001'
MQDMHO_CURRENT_VERSION	1	X'00000001'

Opcje usuwania uchwytu komunikatu

Tabela 145. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDMHO_NONE	0	X'00000000'

MQDMPO_ * (Usuń opcje i strukturę właściwości komunikatu)

Usuń strukturę opcji właściwości komunikatu

Tabela 146. Struktury stałych

Nazwa	Struktura
Identyfikator struktury MQDMPO_STRUC_ID	"DMPO"
MQDMPO_STRUC_ID_ARRAY	'D','M','P','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 147. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQDMPO_VERSION_1	1	X'00000001'
MQDMPO_CURRENT_VERSION	1	X'00000001'

Opcje usuwania właściwości komunikatu

Tabela 148. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDMPO_DEL_FIRST	0	X'00000000'
MQDMPO_DEL_PROP_UNDER_CURSOR,	1	X'00000001'
MQDMPO_BRAK	0	X'00000000'

MQDNSWLM_ * (WLM DNS)

Tabela 149. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDNSWLM_NO	0	X'00000000'
MQDNSWLM_TAK	1	X'00000001'

MQDT_ * (typy docelowe)

Tabela 150. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDT_APPL	1	X'00000001'
MQDT_BROKER	2	X'00000002'

MQDXP_ * (struktura parametru wyjścia konwersji)

Tabela 151. Struktury stałych	
Nazwa	**Struktura**
MQDXP_STRUC_ID	"DXP¬"
MQDXP_STRUC_ID_ARRAY	'D','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 152. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQDXP_VERSION_1	1	X'00000001'
MQDXP_VERSION_2	2	X'00000002'
MQDXP_CURRENT_VERSION	2	X'00000002'

MQEC_ * (wartości sygnału)

Tabela 153. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQEC_MSG_ODEBRANY	2	X'00000002'
MQEC_WAIT_INTERVAL_EXPIRED:	3	X'00000003'
MQEC_WAIT_ANULOWANA	4	X'00000004'
MQEC_Q_MGR_QUIESCING,	5	X'00000005'
MQEC_CONNECTION_QUIESCING,	6	X'00000006'

MQEI_ * (Utrata ważności)

Tabela 154. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEI_UNLIMITED,	-1	X'FFFFFFFF'

MQENC_ * (Kodowanie)

MQENC_ * (Kodowanie)

Tabela 155. Wartości stałych według platformy

Nazwa	Platforma	Wartość dziesiętna	Wartość szesnastkowa
RODZIMA MQENC	IBM i	273	X'00000111'
	Linux	546	X'00000222'
	Linux na platformie SPARC	273	X'00000111'
	Linux na platformie x86	546	X'00000222'
	AIX and Linux	273	X'00000111'
	Windows	546	X'00000222'
	Micro Focus COBOL na platformie Windows	17	X'00000011'
	z/OS	785	X'00000311'

MQENC_ * (maski kodowania)

Tabela 156. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQENC_INTEGER_MASK (maska_liczba_całk)	15	X'0000000F'
MQENC_DECIMAL_MASK	240	X'000000F0'
MQENC_FLOAT_MASK	3840	X'00000F00'
MASKA_ZAREZERWOWANA_MQENC_	-4096	X'FFFFF000'

MQENC_ * (kodowanie dla binarnych liczb całkowitych)

Tabela 157. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQENC_INTEGER_UNDEFINED,	0	X'00000000'
MQENC_INTEGER_NORMAL	1	X'00000001'
MQENC_INTEGER_REVERSED,	2	X'00000002'

MQENC_ * (Kodowanie dla upakowanych dziesiętnych liczb całkowitych)

Tabela 158. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQENC_DECIMAL_UNDEFINED,	0	X'00000000'
MQENC_DECIMAL_NORMAL	16	X'00000010'
MQENC_DECIMAL_REVERSED (odwrócone)	32	X'00000020'

MQENC_ * (kodowanie liczb zmiennopozycyjnych)

Tabela 159. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (struktura nagłówka i flagi formatu komendy osadzonej)

Struktura nagłówka osadzonego formatu komendy

Tabela 160. Struktury stałych

Nazwa	Struktura
ID_STRUKTURY_MQEPH_STRUCT	"EPH¬"
MQEPH_STRUC_ID_ARRAY	'E','P','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 161. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEPH_STRUC_LENGTH_FIXED	68	X'00000044'
MQEPH_VERSION_1	1	X'00000001'
MQEPH_CURRENT_VERSION	1	X'00000001'

Flagi nagłówka osadzonego formatu komendy

Tabela 162. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEPH_BRAK	0	X'00000000'
MQEPH_CCSID_EMBEDDED	1	X'00000001'

MQET_ * (typy zmiany znaczenia formatu komendy)

Tabela 163. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQET_MQSC	1	X'00000001'

MQEVO_ * (źródła zdarzeń w formacie komendy)

Tabela 164. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQEVO_INNY	0	X'00000000'
MQEVO_CONSOLE	1	X'00000001'
MQEVO_INIT,	2	X'00000002'
MQEVO_MSG	3	X'00000003'

Tabela 164. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQEVO_MQSET	4	X'00000004'
MQEVO_INTERNAL	5	X'00000005'
MQEVO_MQSUB	6	X'00000006'
MQEVO_CTLMSG	7	X'00000007'
MQEVO_REST	8	X'00000008'

MQEVR_ * (Rejestrowanie zdarzeń w formacie komendy)

Tabela 165. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQEVR_DISABLED (wyłączone)	0	X'00000000'
MQEVR_ENABLED,	1	X'00000001'
MQEVR_EXCEPTION,	2	X'00000002'
MQEVR_NO_DISPLAY	3	X'00000003'

MQEXPI_ * (odstęp czasu skanowania utraty ważności)

Tabela 166. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQEXPI_OFF	0	X'00000000'

MQFB_ * (Wartości zapisu czynności)

Tabela 167. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQFB_BRAK	0	X'00000000'
MQFB_SYSTEM_FIRST	1	X'00000001'
MQFB_QUIT	256	X'00000100'
MQFB_UTR_WAŻN.	258	X'00000102'
MQFB_COA	259	X'00000103'
MQFB_COD	260	X'00000104'
ZAKOŃCZONE_KANAŁU_MQFB_CHANN_XX_ENCODE_CASE_CAPS_LOCK_OFF	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'
BŁĄD 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'

Tabela 167. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
DZIAŁANIA MQFB_MAKSIMUM	282	X'0000011A'
MQFB_NOT_FORWARDED	283	X'0000011B'
MQFB_NIEDELIVERED	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_NEGATIVE	292	X'00000124'
MQFB_DATA_LENGTH_TOO_BIG	293	X'00000125'
Przepełnienie buforu MQFB_BUFFER_OVERFLOW	294	X'00000126'
MQFB_LENGTH_OFF_BY_ONE	295	X'00000127'
MQFB_IIH_BŁĄD	296	X'00000128'
MQFB_NOT_AUTHORIZED_FOR_IMS (MQFB_NIE_AUTHORITY)	298	X'0000012A'
BŁĄD MQFB_IMS_ERROR	300	X'0000012C'
MQFB_IMS_FIRST	301	X'0000012D'
MQFB_IMS_LAST	399	X'0000018F'
MQFB_CICS_INTERNAL_ERROR (Błąd interfejsu CICS)	401	X'00000191'
MQFB_CICS_NOT_AUTHORIZED	402	X'00000192'
MQFB_CICS_BRIDGE_FAILURE	403	X'00000193'
BŁĄD MQFB_CICS_CORREL_ID_ERROR	404	X'00000194'
BŁĄD MQFB_CICS_CCSID_ERROR	405	X'00000195'
BŁĄD WYWOŁANIA MQFB_CICS_ENCODING_ERROR	406	X'00000196'
BŁĄD MQFB_CICS_CIH_ERROR	407	X'00000197'
BŁĄD MQFB_CICS_UOW_ERROR	408	X'00000198'
BŁĄD MQFB_CICS_COMMAREA_ERROR	409	X'00000199'
MQFB_CICS_APPL_NOT_STARTED (nie uruchomiono)	410	X'0000019A'
MQFB_CICS_APPL_ABENDED	411	X'0000019B'
BŁĄD MQFB_CICS_DLQ_ERROR	412	X'0000019C'
MQFB_CICS_UOW_BACKED_OUT	413	X'0000019D'
Publikacje MQFB_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'

Tabela 167. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQFB_APPL_FIRST	65536	X'00010000'
MQFB_APPL_LAST	999999999	X'3B9AC9FF'

MQFC_ * (Opcje wymuszania formatu komendy)

Tabela 168. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQFC_TAK	1	X'00000001'
MQFC_NO	0	X'00000000'

MQFMT_ * (formaty)

Tabela 169. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQFMT_BRAK	"¬¬¬¬¬¬¬¬"
MQFMT_ADMIN,	"MQADMIN¬"
MQFMT_CHANNEL_COMPLETED	"MQCHCOM¬"
MQFMT_CICS	"MQCICS¬¬"
MQFMT_COMMAND_1	"MQCMD1¬¬"
MQFMT_COMMAND_2	"MQCMD2¬¬"
MQFMT_DEAD_LETTER_HEADER	"MQDEAD¬¬"
Usługa MQFMT_DIST_HEADER	"MQHDIST¬"
MQFMT_EMBEDDED_PCF	"MQHEPCF¬"
MQFMT_EVENT, ZDARZENIE	"MQEVENT¬"
MQFMT_IMS,	"MQIMS¬¬¬"
MQFMT_IMS_VAR_STRING (łańcuch zmiennych)	"MQIMSVS¬"
MQFMT_MD_EXTENSION	"MQHMDE¬¬"
MQFMT_PCF,	"MQPCF¬¬¬"
MQFMT_REF_MSG_HEADER	"MQHREF¬¬"
ZMQFMT_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	'¬','¬','¬','¬','¬','¬','¬','¬'
Tablica 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','¬','¬'

Tabela 169. Nazwy i wartości stałych (kontynuacja)	
Nazwa	**Wartość**
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','¬','¬'
Tabela 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','¬','¬'
Tablica łańcuchów 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','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQFUN_ * (typy funkcji aplikacji)

Tabela 170. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQFUN_TYPE_UNKNOWN	0	X'00000000'
MQFUN_TYPE_JVM	1	X'00000001'
PROGRAM TYPU FUNKCJI MQFUNCTION	2	X'00000002'
PROCEDURA TYPU FUNKCJI MQFUNCTION	3	X'00000003'
MQFUN_TYPE_USERDEF	4	X'00000004'
KOMENDA MQFUN_TYPE_COMMAND	5	X'00000005'

MQGA_ * (Selektory atrybutów grupy)

Tabela 171. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQGA_FIRST	8001	X'00001F41'
MQGA_LAST	9000	X'00002328'

MQGACF_ * (typy parametrów grupy formatu komendy)

Tabela 172. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQGACF_FIRST	8001	X'00001F41'

Tabela 172. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQGACF_COMMAND_CONTEXT	8001	X'00001F41'
MQGACF_COMMAND_DATA	8002	X'00001F42'
MQGACF_TRACE_ROUTE (trasa śledzenia MQGACF)	8003	X'00001F43'
OPERACJA MQGACF	8004	X'00001F44'
AKTYWNOŚĆ MQGACF	8005	X'00001F45'
MQGACF_EMBEDDED_MQMD	8006	X'00001F46'
MQGACF_MESSAGE	8007	X'00001F47'
MQGACF_MQMD	8008	X'00001F48'
MQGACF_VALUE_NAMING (nazwa wartości MQGACF)	8009	X'00001F49'
LICZNIK MQGACF_Q_ACTING_DATA	8010	X'00001F4A'
MQGACF_Q_STATISTICS_DATA (dane statyczne)	8011	X'00001F4B'
MQGACF_CHL_STATISTICS_DATA	8012	X'00001F4C'
MQGACF_LAST_UŻYWANE	8012	X'00001F4C'

MQGI_ * (identyfikator grupy)

Tabela 173. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQGI_NONE	X'00...00' (24 wartości null)
MQGI_NONE_ARRAY	'\0','\0',... (24 wartości null)

MQGMO_ * (pobranie opcji i struktury komunikatu)

Pobierz strukturę opcji komunikatu

Tabela 174. Struktury stałych	
Nazwa	**Struktura**
MQGMO_ID_struktury	"GMO¬"
MQGMO_STRUC_ID_ARRAY	'G','M','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 175. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQGMO_VERSION_1	1	X'00000001'
MQGMO_VERSION_2	2	X'00000002'
MQGMO_VERSION_3	3	X'00000003'
MQGMO_VERSION_4	4	X'00000004'
MQGMO_CURRENT_VERSION	4	X'00000004'

Opcje pobierania wiadomości

Tabela 176. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'
BLOKADA MQGMO_LOCK	512	X'00000200'
MQGMO_UNLOCK (odblokowanie MQGMO)	1024	X'00000400'
MQGMO_ACCEPT_TRUNCATED_MSG	64	X'00000040'
MQGMO_CONVERT	16384	X'00004000'
MQGMO_LOGICAL_ORDER (KOLEJNA_KOLEJNA_STRUKTURA)	32768	X'00008000'
MQGMO_COMPLETE_MSG	65536	X'00010000'
MQGMO_ALL_MSGS_AVAILABLE	131072	X'00020000'
MQGMO_ALL_SEGMENTS_AVAILABLE	262144	X'00040000'
MQGMO_MARK_BROWSE_HANDLE	1048576	X'00100000'
MQGMO_MARK_BROWSE_CO_OP	2097152	X'00200000'
MQGMO_UNMARK_BROWSE_CO_OP	4194304	X'00400000'
MQGMO_UNMARK_BROWSE_HANDLE	8388608	X'00800000'
MQGMO_UNMARKED_BROWSE_MSG	16777216	X'01000000'
MQGMO_PROPERTIES_FORCE_MQRFH2	33554432	X'02000000'
MQGMO_NO_PROPERTIES	67108864	X'04000000'
MQGMO_PROPERTIES_IN_HANDLE	134217728	X'08000000'
MQGMO_PROPERTIES_COMPATIBILITY	268435456	X'10000000'
MQGMO_PROPERTIES_AS_Q_DEF	0	X'00000000'
MQGMO_NONE	0	X'00000000'

MQGS_ * (Status grupy)

Tabela 177. Nazwy i wartości stałych	
Nazwa	**Wartość**
GRUPA MQGS_NOT_IN_	'¬'

Tabela 177. Nazwy i wartości stałych (kontynuacja)	
Nazwa	**Wartość**
MQGS_MSG_IN_GROUP	'G'
MQGS_LAST_MSG_IN_GROUP	'L'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQHA_ * (Obsługa selektorów)

Tabela 178. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQHA_FIRST	4001	X'00000FA1'
MQHA_BAG_HANDLE	4001	X'00000FA1'
MQHA_LAST_USED (OSTATNI_UŻYTY)	4001	X'00000FA1'
MQHA_LAST	6000	X'00001770'

MQHB_ * (uchwyty wielozbioru)

Tabela 179. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQHB_UNUSABLE_HBAG	-1	X'FFFFFFFF'
MQHB_NONE	-2	X'FFFFFFFE'

MQHC_ * (uchwyty połączenia)

Tabela 180. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
ZMQHC_DEF_HCONN	0	X'00000000'
MQHC_UNUSABLE_HCONN	-1	X'FFFFFFFF'
MQHC_UNASSOCIATED_HCONN	-3	X'FFFFFFFD'

MQHM_ * (uchwyt komunikatu)

Tabela 181. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQHM_UNUSABLE_HMSG	-1	X'FFFFFFFF'
MQHM_NONE	0	X'00000000'

MQHO_ * (uchwyt obiektu)

Tabela 182. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQHO_UNUSABLE_HOBJ	-1	X'FFFFFFFF'
MQHO_NONE	0	X'00000000'

MQHSTATE_ * (stany uchwytu formatu komendy)

Tabela 183. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQHSTATE_INACTIVE (NIEAKTYWNE)	0	X'00000000'
MQHSTATE_ACTIVE (aktywne MQHSTATE)	1	X'00000001'

MQIA_ * (Selektory Atrybutów Całkowitoliczbowych)

Tabela 184. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_ACCOUNTING_CONN_OVERRIDE	136	X'00000088'
MQIA_ACCOUNTING_INTERVAL	135	X'00000087'
MQIA_ACCOUNTING_MQI	133	X'00000085'
MQIA_ACCOUNTING_Q	134	X'00000086'
MQIA_ACTIVE_CHANNELS	100	X'00000064'
MQIA_ACTIVITY_CONN_OVERRIDE	239	X'000000EF'
MQIA_ACTIVITY_RECORDING	138	X'0000008A'
MQIA_ACTIVITY_TRACE	240	X'000000F0'
MQIA_ADOPTNEWMCA_CHECK	102	X'00000066'
MQIA_ADOPTNEWMCA_INTERVAL	104	X'00000068'
MQIA_ADOPTNEWMCA_TYPE	103	X'00000067'
MQIA_ADOPT_CONTEXT	260	X'00000104'
MQIA_ADVANCED_CAPABILITY	273	X'00000111'
MQIA_AMQP_CAPABILITY	265	X'00000109'
MQIA_APPL_TYPE	1	X'00000001'
MQIA_ARCHIVE	60	X'0000003C'
MQIA_AUTHENTICATION_FAIL_DELAY	259	X'00000103'
MQIA_AUTHENTICATION_METHOD	266	X'0000010A'
MQIA_AUTH_INFO_TYPE	66	X'00000042'
MQIA_AUTHORITY_EVENT	47	X'0000002F'
MQIA_AUTO_REORG_INTERVAL	174	X'000000AE'
MQIA_AUTO_REORGANIZATION	173	X'000000AD'
MQIA_BACKOUT_THRESHOLD	22	X'00000016'
MQIA_BASE_TYPE	193	X'000000C1'
MQIA_BATCH_INTERFACE_AUTO	86	X'00000056'
MQIA_BRIDGE_EVENT	74	X'0000004A'
MQIA_CF_LEVEL	70	X'00000046'
MQIA_CF_RECOVER	71	X'00000047'
MQIA_CHANNEL_AUTO_DEF	55	X'00000037'
MQIA_CHANNEL_AUTO_DEF_EVENT	56	X'00000038'
MQIA_CHANNEL_EVENT	73	X'00000049'
MQIA_CHECK_CLIENT_BINDING	258	X'00000102'

Tabela 184. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_CHECK_LOCAL_BINDING	257	X'00000101'
MQIA_CHINIT_ADAPTERS	101	X'00000065'
MQIA_CHINIT_CONTROL	119	X'00000077'
MQIA_CHINIT_DISPATCHERS	105	X'00000069'
MQIA_CHINIT_TRACE_AUTO_START	117	X'00000075'
MQIA_CHINIT_TRACE_TABLE_SIZE	118	X'00000076'
MQIA_CLUSTER_OBJECT_STATE	256	X'00000100'
MQIA_CLUSTER_PUB_ROUTE	255	X'000000FF'
MQIA_CLUSTER_Q_TYPE	59	X'0000003B'
MQIA_CLUSTER_WORKLOAD_LENGTH	58	X'0000003A'
MQIA_CLWL_MRU_CHANNELS	97	X'00000061'
MQIA_CLWL_Q_RANK	95	X'0000005F'
MQIA_CLWL_Q_PRIORITY	96	X'00000060'
MQIA_CLWL_USEQ	98	X'00000062'
MQIA_CMD_SERVER_AUTO	87	X'00000057'
MQIA_CMD_SERVER_CONTROL	120	X'00000078'
MQIA_CMD_SERVER_CONVERT_MSG	88	X'00000058'
MQIA_CMD_SERVER_DLQ_MSG	89	X'00000059'
MQIA_CODED_CHAR_SET_ID	2	X'00000002'
MQIA_COMM_EVENT	232	X'000000E8'
MQIA_COMMAND_EVENT	99	X'00000063'
MQIA_COMMAND_LEVEL	31	X'0000001F'
MQIA_CONFIGURATION_EVENT	51	X'00000033'
MQIA_CPI_LEVEL	27	X'0000001B'
MQIA_CURRENT_Q_DEPTH	3	X'00000003'
MQIA_DEF_BIND	61	X'0000003D'
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	250	X'000000FA'
MQIA_DEF_INPUT_OPEN_OPTION	4	X'00000004'
MQIA_DEF_PERSISTENCE	5	X'00000005'
MQIA_DEF_PRIORITY	6	X'00000006'
MQIA_DEF_PUT_RESPONSE_TYPE	184	X'000000B8'
MQIA_DEF_READ_AHEAD	188	X'000000BC'
MQIA_DEFINITION_TYPE	7	X'00000007'
MQIA_DISPLAY_TYPE	262	X'00000106'
MQIA_DIST_LISTS	34	X'00000022'
MQIA_DNS_WLM	106	X'0000006A'
MQIA_DURABLE_SUB	175	X'000000AF'
MQIA_EXPIRY_INTERVAL	39	X'00000027'
MQIA_FIRST	1	X'00000001'

Tabela 184. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIA_GROUP_UR	221	X'000000DD'
MQIA_HARDEN_GET_BACKOUT	8	X'00000008'
MQIA_HIGH_Q_DEPTH	36	X'00000024'
MQIA_IGQ_PUT_AUTHORITY	65	X'00000041'
MQIA_INDEX_TYPE	57	X'00000039'
MQIA_INHIBIT_EVENT	48	X'00000030'
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_KEY_REUSE_COUNT	267	X'0000010B'
MQIA_LAST	2000	X'000007D0'
MQIA_LAST_USED	267	X'0000010B'
MQIA_LDAP_AUTHORMD	263	X'00000107'
MQIA_LDAP_NESTGRP	264	X'00000108'
MQIA_LDAP_SECURE_COMM	261	X'00000105'
MQIA_LISTENER_PORT_NUMBER	85	X'00000055'
MQIA_LISTENER_TIMER	107	X'0000006B'
MQIA_LOGGER_EVENT	94	X'0000005E'
MQIA_LU62_CHANNELS	108	X'0000006C'
MQIA_LOCAL_EVENT	49	X'00000031'
MQIA_MSG_MARK_BROWSE_INTERVAL	68	X'00000044'
MQIA_MAX_CHANNELS	109	X'0000006D'
MQIA_MAX_CLIENTS	172	X'000000AC'
MQIA_MAX_GLOBAL_LOCKS	83	X'00000053'
MQIA_MAX_HANDLES	11	X'0000000B'
MQIA_MAX_LOCAL_LOCKS	84	X'00000054'
MQIA_MAX_MSG_LENGTH	13	X'0000000D'
MQIA_MAX_OPEN_Q	80	X'00000050'
MQIA_MAX_PRIORITY	14	X'0000000E'
MQIA_MAX_PROPERTIES_LENGTH	192	X'000000C0'
MQIA_MAX_Q_DEPTH	15	X'0000000F'
MQIA_MAX_Q_TRIGGERS	90	X'0000005A'
MQIA_MAX_RECOVERY_TASKS	171	X'000000AB'
MQIA_MAX_UNCOMMITTED_MSGS	33	X'00000021'
MQIA_MCAST_BRIDGE	233	X'000000E9'
MQIA_MONITOR_INTERVAL	81	X'00000051'

Tabela 184. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
MQIA_NPM_DELIVERY	196	X'000000C4'
MQIA_OPEN_INPUT_COUNT	17	X'00000011'
MQIA_OPEN_OUTPUT_COUNT	18	X'00000012'
MQIA_OUTBOUND_PORT_MAX	140	X'0000008C'
MQIA_OUTBOUND_PORT_MIN	110	X'0000006E'
MQIA_PAGESET_ID	62	X'0000003E'
MQIA_PERFORMANCE_EVENT	53	X'00000035'
MQIA_PLATFORM	32	X'00000020'
MQIA_PM_DELIVERY	195	X'000000C3'
MQIA_PROPERTY_CONTROL	190	X'000000BE'
MQIA_PROT_POLICY_CAPABILITY	251	X'000000FB'
MQIA_PROXY_SUB	199	X'000000C7'
MQIA_PUB_COUNT	215	X'000000D7'
MQIA_PUB_SCOPE	219	X'000000DB'
MQIA_PUBSUB_CLUSTER	249	X'000000F9'
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	206	X'000000CE'
MQIA_PUBSUB_MODE	187	X'000000BB'
MQIA_PUBSUB_NP_MSG	203	X'000000CB'
MQIA_PUBSUB_NP_RESP	205	X'000000CD'
MQIA_PUBSUB_SYNC_PT	207	X'000000CF'
MQIA_Q_DEPTH_HIGH_EVENT	43	X'0000002B'
MQIA_Q_DEPTH_HIGH_LIMIT	40	X'00000028'
MQIA_Q_DEPTH_LOW_EVENT	44	X'0000002C'
MQIA_Q_DEPTH_LOW_LIMIT	41	X'00000029'
MQIA_Q_DEPTH_MAX_EVENT	42	X'0000002A'
MQIA_Q_SERVICE_INTERVAL	54	X'00000036'
MQIA_Q_SERVICE_INTERVAL_EVENT	46	X'0000002E'
MQIA_Q_TYPE	20	X'00000014'
MQIA_Q_USERS	82	X'00000052'
MQIA_QMGR_CFCONLOS	245	X'000000F5'

Tabela 184. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
MQIA_QMOPT_LOG_ERROR_MSGS	160	X'000000A0'
MQIA_QMOPT_LOG_INFO_MSGS	158	X'0000009E'
MQIA_QMOPT_LOG_REORG_MSGS	163	X'000000A3'
MQIA_QMOPT_LOG_SYSTEM_MSGS	164	X'000000A4'
MQIA_QMOPT_LOG_WARNING_MSGS	159	X'0000009F'
MQIA_QMOPT_TRACE_COMMS	166	X'000000A6'
MQIA_QMOPT_TRACE_CONVERSION	168	X'000000A8'
MQIA_QMOPT_TRACE_REORG	167	X'000000A7'
MQIA_QMOPT_TRACE_MQI_CALLS	165	X'000000A5'
MQIA_QMOPT_TRACE_SYSTEM	169	X'000000A9'
MQIA_QSG_DISP	63	X'0000003F'
MQIA_READ_AHEAD	189	X'000000BD'
MQIA_RECEIVE_TIMEOUT	111	X'0000006F'
MQIA_RECEIVE_TIMEOUT_MIN	113	X'00000071'
MQIA_RECEIVE_TIMEOUT_TYPE	112	X'00000070'
MQIA_REMOTE_EVENT	50	X'00000032'
MQIA_RETENTION_INTERVAL	21	X'00000015'
MQIA_REVERSE_DNS_LOOKUP	254	X'000000FE'
MQIA_SCOPE	45	X'0000002D'
MQIA_SECURITY_CASE	141	X'0000008D'
MQIA_SERVICE_CONTROL	139	X'0000008B'
MQIA_SERVICE_TYPE	121	X'00000079'
MQIA_SHAREABILITY	23	X'00000017'
MQIA_SHARED_Q_Q_MGR_NAME	77	X'0000004D'
MQIA_SSL_EVENT	75	X'0000004B'
MQIA_SSL_FIPS_REQUIRED	92	X'0000005C'
MQIA_SSL_RESET_COUNT	76	X'0000004C'
MQIA_SSL_TASKS	69	X'00000045'

Tabela 184. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
MQIA_TIME_SINCE_RESET	35	X'00000023'
MQIA_TOPIC_DEF_PERSISTENCE	185	X'000000B9'
MQIA_TOPIC_NODE_COUNT	253	X'000000FD'
MQIA_TOPIC_TYPE	208	X'000000D0'
MQIA_TRACE_ROUTE_RECORDING	137	X'00000089'
MQIA_TREE_LIFE_TIME	183	X'000000B7'
MQIA_TRIGGER_CONTROL	24	X'00000018'
MQIA_TRIGGER_DEPTH	29	X'0000001D'
MQIA_TRIGGER_INTERVAL	25	X'00000019'
MQIA_TRIGGER_MSG_PRIORITY	26	X'0000001A'
MQIA_TRIGGER_TYPE	28	X'0000001C'
MQIA_TRIGGER_RESTART	91	X'0000005B'
MQIA_USAGE	12	X'0000000C'
MQIA_USE_DEAD_LETTER_Q	234	X'000000EA'
MQIA_USER_LIST	2000	X'000007D0'
MQIA_WILDCARD_OPERATION	216	X'000000D8'
MQIA_XR_CAPABILITY	243	X'000000F3'

MQIACF_ * (typy parametrów całkowitoliczbowych formatu komendy)

Tabela 185. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_REPLACE	1006	X'000003EE'
MQIACF_PURGE	1007	X'000003EF'
MQIACF_QUIESCE	1008	X'000003F0'
TRYB_MQIACF	1008	X'000003F0'
MQIACF_ALL	1009	X'000003F1'
MQIACF_EVENT_APPL_TYPE	1010	X'000003F2'
MQIACF_EVENT_ORIGIN	1011	X'000003F3'
ID_parametru_MQIACF_PARAMETER_ID	1012	X'000003F4'
MQIACF_ERROR_ID (Identyfikator błędu MQIACF)	1013	X'000003F5'
MQIACF_ERROR_IDENTIFIER	1013	X'000003F5'
MQIACF_SELECTOR	1014	X'000003F6'
MQIACF_CHANNEL_ATTRS	1015	X'000003F7'
MQIACF_XX_ENCODE_CASE_ONE typ_obiektu	1016	X'000003F8'
MQIACF_ESCAPE_TYPE	1017	X'000003F9'
MQIACF_ERROR_OFFSET	1018	X'000003FA'
MQIACF_AUTH_INFO_ATTRS	1019	X'000003FB'
KWALIFIKATOR PRZYCZYNY MQIACF_REASON_QUALIFIER	1020	X'000003FC'
MQIACF_COMMAND (komenda MQIACF)	1021	X'000003FD'
OPCJE MQIACF_OPEN_OPTIONS	1022	X'000003FE'
MQIACF_OPEN_TYPE	1023	X'000003FF'
ID_PROCESu_MQIACF	1024	X'00000400'
MQIACF_THREAD_ID (Identyfikator wątku MQIACF)	1025	X'00000401'
MQIACF_Q_STATUS_ATTRS	1026	X'00000402'
MQIACF_UNCOMMITTED_MSGS	1027	X'00000403'
MQIACF_HANDLE_STATE	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_TYP_MOSTU	1073	X'00000431'
MQIACF_INQUIRY	1074	X'00000432'
ODSTĘP CZASU OCZEKIWANIA MQIACF_WAIT_INTERVAL	1075	X'00000433'
OPCJE MQIACF	1076	X'00000434'
MQIACF_OPCJE_BROKERA	1077	X'00000435'
MQIACF_REFRESH_TYPE	1078	X'00000436'
NUMER KOLEJKI MQIACF_SEQUENCE_NUMBER	1079	X'00000437'
DANE MQIACF_INTEGER_DATA	1080	X'00000438'
Opcje MQIACF_REGISTRATION_OPTIONS	1081	X'00000439'
OPCJE MQIACF_PUBLIC	1082	X'0000043A'
MQIACF_CLUSTER_INFO	1083	X'0000043B'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_Q_MGR_DEFINITION_TYPE	1084	X'0000043C'
MQIACF_Q_MGR_TYPE	1085	X'0000043D'
MQIACF_DZIAŁANIE	1086	X'0000043E'
MQIACF_ZAWIEŚ	1087	X'0000043F'
MQIACF_BROKER_COUNT (MQIACF)	1088	X'00000440'
Liczba operacji MQIACF_APPL_COUNT	1089	X'00000441'
LICZBA OPERACJI MQIACF_ANONYMOUS_COUNT	1090	X'00000442'
OPCJE MQIACF_REG_REGTIONS	1091	X'00000443'
OPCJE MQIACF_DELETE_	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'
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'
STATUS MQIACF_Q_STATUS	1105	X'00000451'
MQIACF_SECURITY_TYPE	1106	X'00000452'
MQIACF_CONNECTION_ATTRS	1107	X'00000453'
OPCJE MQIACF_CONNECT_	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_AUTHORIZATION_LIST (lista uprawnień MQIACF)	1115	X'0000045B'
MQIACF_AUTH_ADD_AUTHS	1116	X'0000045C'
MQIACF_AUTH_REMOVE_AUTHS	1117	X'0000045D'
MQIACF_ENTITY_TYPE	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'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_QSG_DISPS	1126	X'00000466'
STAN MQIACF_UOW_STATE	1128	X'00000468'
MQIACF_SECURITY_ITEM	1129	X'00000469'
MQIACF_CF_STRUC_STATUS	1130	X'0000046A'
MQIACF_UOW_TYPE	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'
MQIACF_CF_STRUC_SIZE_UŻYWANE	1141	X'00000475'
MQIACF_CF_STRUC_ENTRIES_MAX	1142	X'00000476'
MQIACF_CF_STRUC_ENTRIES_UŻYWANE	1143	X'00000477'
MQIACF_CF_STRUC_BACKUP_SIZE	1144	X'00000478'
MQIACF_MOVE_TYPE (typ przeniesienia MQ)	1145	X'00000479'
MQIACF_MOVE_TYPE_MOVE (przenoszenie typu MQIACF)	1146	X'0000047A'
MQIACF_MOVE_TYPE_ADD	1147	X'0000047B'
MQIACF_Q_MGR_NUMBER	1148	X'0000047C'
MQIACF_Q_MGR_STATUS	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'
MQIACF_USAGE_TYPE (typ MQIACF)	1157	X'00000485'
MQIACF_BUFFER_POOL_ID	1158	X'00000486'
MQIACF_USAGE_TOTAL_PAGES	1159	X'00000487'
STRONY MQIACF_USAGE_UNUSED_PAGES	1160	X'00000488'
MQIACF_USAGE_PERSIST_STRONY	1161	X'00000489'
MQIACF_USAGE_NONPERSIST_PAGES	1162	X'0000048A'
MQIACF_USAGE_RESTART_EXTENTS	1163	X'0000048B'
MQIACF_USAGE_EXPAND_COUNT,	1164	X'0000048C'
STATUS STRONY MQIACF_PAGESET_STATUS	1165	X'0000048D'
MQIACF_USAGE_TOTAL_BUFFERS	1166	X'0000048E'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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 (mqiacf_pula_buforów)	1170	X'00000492'
Liczba operacji MQIACF_MOVE_COUNT	1171	X'00000493'
MQIACF_EXPIRY_Q_COUNT	1172	X'00000494'
MQIACF_CONFIGURATION_OBJECTS	1173	X'00000495'
MQIACF_XX_ENCODE_CASE_ONE konfiguracja_zdarzenia	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'
MQIACF_SYSP_OUT_BUFFER_SIZE	1180	X'0000049C'
MQIACF_SYSP_OUT_BUFFER_COUNT (MQIACF)	1181	X'0000049D'
MQIACF_SYSP_ARCHIVE	1182	X'0000049E'
MQIACF_SYSP_DUAL_ACTIVE (aktywne)	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_przodem	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'
Komenda 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'
KOD MQIACF_SYSP_ROUTING_CODE	1196	X'000004AC'
MQIACF_SYSP_SMF_KONTOWANIE	1197	X'000004AD'
MQIACF_SYSP_SMF_STATS	1198	X'000004AE'
MQIACF_SYSP_SMF_INTERVAL	1199	X'000004AF'
KLASA 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'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_SYSP_BLOCK_SIZE	1206	X'000004B6'
MQIACF_SYSP_CATALOG	1207	X'000004B7'
MQIACF_SYSP_COMPACT	1208	X'000004B8'
MQIACF_SYSP_ALLOC_PRIMARY (podstawowy)	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 (MQIACF_SYSP_LOG_COPY	1216	X'000004C0'
MQIACF_SYSP_LOG_USED	1217	X'000004C1'
MQIACF_SYSP_LOG_SUSPEND	1218	X'000004C2'
STATUS PRZESUNIĘCIA 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_TIME_INDICATOR	1226	X'000004CA'
MQIACF_OLDEST_MSG_AGE	1227	X'000004CB'
OPCJE MQIACF_AUTH_OPCJE	1228	X'000004CC'
MQIACF_Q_MGR_STATUS_ATTRS	1229	X'000004CD'
LICZBA POŁĄCZEŃ 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'
DZIAŁANIA MQIACF_RECORDED_ACTIVITIES	1235	X'000004D3'
DZIAŁANIA MQIACF_MAX_ITIES	1236	X'000004D4'
MQIACF_DISCONTINUITY_COUNT	1237	X'000004D5'
MQIACF_ROUTE_AKUMULACJA	1238	X'000004D6'
MQIACF_ROUTE_DELIVERY	1239	X'000004D7'
MQIACF_OPERATION_TYPE (typ operacji MQ)	1240	X'000004D8'
MQIACF_BACKOUT_COUNT	1241	X'000004D9'
KOD MQIACF_COMP_CODE	1242	X'000004DA'
MQIACF_ENCODING	1243	X'000004DB'
MQIACF_WAŻNOŚĆ	1244	X'000004DC'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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 (długość tabeli MQIACF)	1251	X'000004E3'
MQIACF_PERSISTENCE	1252	X'000004E4'
PRIORYTET_MQIACF	1253	X'000004E5'
KOD_REASON_CODE MQIACF_REASON_CODE	1254	X'000004E6'
MQIACF_RAPORT	1255	X'000004E7'
MQIACF_XX_ENCODE_CASE_ONE wersja	1256	X'000004E8'
MQIACF_UNRECORDED_ACTIVITIES MQIACF_UNRECORDED_ACTIVITIES	1257	X'000004E9'
MQIACF_MONITORING	1258	X'000004EA'
MQIACF_ROUTE_FORWARDING	1259	X'000004EB'
MQIACF_STATUS USŁUGI	1260	X'000004EC'
MQIACF_Q_TYPES	1261	X'000004ED'
OBSŁUGA MQIACF_USER_ID_SUPPORT	1262	X'000004EE'
WERSJA INTERFEJSU 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'
WŁAŚCIWOŚCI MQIACF_PUBSUB_PROPERTIES	1271	X'000004F7'
MQIACF_XX_ENCODE_CASE_ONE klasa_docelowego	1273	X'000004F9'
MQIACF_DURABLE_SUBSCRIPTION	1274	X'000004FA'
MQIACF_SUBSCRIPTION_SCOPE	1275	X'000004FB'
ID_UŻYTKOWNIKA MQIACF_VARIABLE_USER_ID	1277	X'000004FD'
MQIACF_XX_ENCODE_CASE_CAPS_LOCK_ON tylko żądanie	1280	X'00000500'
MQIACF_PUB_PRIORYTET	1283	X'00000503'
MQIACF_SUB_ATTRS	1287	X'00000507'
MQIACF_WILDCARD_SCHEMA (schemat MQIACF)	1288	X'00000508'
MQIACF_SUB_TYPE	1289	X'00000509'
MQIACF_MESSAGE_COUNT (liczba komunikatów MQIACF)	1290	X'0000050A'
MQIACF_Q_MGR_PUBSUB	1291	X'0000050B'
MQIACF_Q_MGR_VERSION	1292	X'0000050C'
MQIACF_SUB_STATUS_ATTRS	1294	X'0000050E'
STATUS TABELI MQIACF_TOPIC_STATUS	1295	X'0000050F'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
OPCJE MQIACF_SUB_OPCJE	1303	X'00000517'
LICZBA OPERACJI MQIACF_PUBLISH_COUNT	1304	X'00000518'
MQIACF_CLEAR_TYPE	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'
MQIACF_PUBSUB_STATUS	1311	X'0000051F'
MQIACF_PS_STATUS_TYPE	1314	X'00000522'
MQIACF_PUBSUB_STATUS_ATTRS	1318	X'00000526'
MQIACF_SELECTOR_TYPE	1321	X'00000529'
MQIACF_MCAST_REL_INDICATOR	1351	X'00000547'
MQIACF_CHLAUTH_TYPE	1352	X'00000548'
MQXR_DIAGNOSTICS_TYPE,	1354	X'0000054A'
MQIACF_CHLAUTH_ATTRS	1355	X'0000054B'
MQIACF_OPERATION_ID (Identyfikator operacji MQIACF)	1356	X'0000054C'
MQIACF_API_CALLER_TYPE	1357	X'0000054D'
MQIACF_API_ENVIRONMENT	1358	X'0000054E'
Szczegóły śledzenia MQIACF_TRACE_DETAIL	1359	X'0000054F'
MQIACF_HOBJ	1360	X'00000550'
MQIACF_CALL_TYPE	1361	X'00000551'
MQIACF_MQCB_OPERACJA	1362	X'00000552'
MQIACF_MQCB_TYPE	1363	X'00000553'
OPCJE MQIACF_MQCB_OPCJE	1364	X'00000554'
OPCJE MQIACF_CLOSE_OPTIONS	1365	X'00000555'
OPERACJA MQIACF_CTL_	1366	X'00000556'
OPCJE MQIACF_GET_OPCJE	1367	X'00000557'
MQIACF_RECS_PRESENT (OBECNOŚĆ)	1368	X'00000558'
MQIACF_KNOWN_DEST_COUNT	1369	X'00000559'
MQIACF_UNKNOWN_DEST_COUNT	1370	X'0000055A'
MQIACF_INVALID_DEST_COUNT	1371	X'0000055B'
MQIACF_RESOLVED_TYPE	1372	X'0000055C'
OPCJE MQIACF_PUT_OPCJE	1373	X'0000055D'

Tabela 185. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIACF_BUFFER_LENGTH,	1374	X'0000055E'
DŁUGOŚĆ DANYCH ŚLEDZENIA MQIACF_TRACE_DATA	1375	X'0000055F'
MQIACF_SMDS_EXPANDST	1376	X'00000560'
DŁUGOŚĆ STRUKTURY MQIACF_STRUC_LENGTH	1377	X'00000561'
MQIACF_ITEM_COUNT (liczba elementów MQIACF)	1378	X'00000562'
MQIACF_UTR_CZAS	1379	X'00000563'
MQIACF_CONNECT_TIME	1380	X'00000564'
MQIACF_DISCONNECT_TIME	1381	X'00000565'
MQIACF_HSUB	1382	X'00000566'
OPCJE MQIACF_SUBRQ_OPTIONS	1383	X'00000567'
MQIACF_XA_RMID	1384	X'00000568'
MQIACF_XA_FLAGS	1385	X'00000569'
MQIACF_XA_RETCODE,	1386	X'0000056A'
MQIACF_XA_HANDLE	1387	X'0000056B'
MQIACF_XA_RETVAL	1388	X'0000056C'
MQIACF_STATUS_TYPE	1389	X'0000056D'
MQIACF_XA_COUNT,	1390	X'0000056E'
MQIACF_SELECTOR_COUNT (liczba MQIACF)	1391	X'0000056F'
MQIACF_SELECTORS	1392	X'00000570'
MQIACF_INTATTR_COUNT	1393	X'00000571'
MQIACF_INTATTRS	1394	X'00000572'
DZIAŁANIE MQIACF_SUBRQ_ACTION	1395	X'00000573'
MQIACF_NUM_PUBS	1396	X'00000574'
MQIACF_POINTER_SIZE	1397	X'00000575'
MQIACF_REMOVE_AUTHREC	1398	X'00000576'
MQIACF_XR_ATTRS	1399	X'00000577'
MQIACF_APPL_FUNCTION_TYPE	1400	X'00000578'
MQIACF_AMQP_ATTRS	1401	X'00000579'
MQIACF_EXPORT_TYPE	1402	X'0000057A'
MQIACF_EXPORT_ATTRS	1403	X'0000057B'
MQIACF_SYSTEM_OBJECTS	1404	X'0000057C'
MQIACF_CONNECTION_SWAP,	1405	X'0000057D'
MQIACF_AMQP_DIAGNOSTICS_TYPE	1406	X'0000057E'
MQIACF_BUFFER_POOL_LOCATION (Położenie puli MQIACF)	1408	X'00000580'
STATUS POŁĄCZENIA MQIACF_LDAP_CONNECTION_STATUS	1409	X'00000581'
MQIACF_SYSP_MAX_ACE_POOL	1410	X'00000582'
MQIACF_PAGECLAS	1411	X'00000583'
MQIACF_AUTH_REC_TYPE	1412	X'00000584'
MQIACF_SYSP_MAX_CONC_OFFLOADS	1413	X'00000585'

Tabela 185. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIACF_SYSP_ZHYPERWRITE	1414	X'00000586'
MQIACF_Q_MGR_STATUS_LOG	1415	X'00000587'
WIELKOŚĆ_DZIENNIKA_ARCHIWUM MQIACF	1416	X'00000588'
MQIACF_MEDIA_XX_ENCODE_CASE_ONE wielkość_dziennika	1417	X'00000589'
MQIACF_RESTART_XX_ENCODE_CASE_ONE wielkość_dziennika	1418	X'0000058A'
MQIACF_REUSABLE_LOG_SIZE	1419	X'0000058B'
MQIACF_LOG_IN_USE	1420	X'0000058C'
MQIACF_LOG_UTILIZATION,	1421	X'0000058D'
Stan MQIACF_IGNORE_STATE	1423	X'0000058F'
MQIACF_MOVABLE_APPL_COUNT (MQIACF)	1424	X'00000590'
MQIACF_APPL_INFO_ATTRS	1425	X'00000591'
MQIACF_APPL_MOVABLE	1426	X'00000592'
MQIACF_REMOTE_QMGR_AKTYWNE	1427	X'00000593'
MQIACF_APPL_INFO_TYPE	1428	X'00000594'
MQIACF_APPL_INFO_APPL	1429	X'00000595'
MQIACF_APPL_INFO_QMGR	1430	X'00000596'
MQIACF_APPL_INFO_LOCAL.	1431	X'00000597'
MQIACF_APPL_IMMOVABLE_COUNT (MQIACF_APPL_IMMABLE_COUNT)	1432	X'00000598'
MQIACF_ZBALANSOWANE	1433	X'00000599'
MQIACF_BALSTATE	1434	X'0000059A'
MQIACF_APPL_IMMOVABLE_REASON	1435	X'0000059B'
MQIACF_DS_ENCRYPTED	1436	X'0000059C'
MQIACF_CUR_Q_FILE_SIZE	1437	X'0000059D'
MQIACF_CUR_MAX_FILE_SIZE	1438	X'0000059E'
MQIACF_BALANCING_TYPE (TYP MQIACF)	1439	X'0000059F'
MQIACF_BALANCING_OPTIONS	1440	X'000005A0'
MQIACF_BALANCING_TIMEOUT (LIMIT_CZASU_OCZEKIWANIA)	1441	X'000005A1'
MQIACF_SYSP_SMF_STAT_TIME_SECS	1442	X'000005A2'
MQIACF_SYSP_SMF_ACCT_TIME_MINS	1443	X'000005A3'
MQIACF_SYSP_SMF_ACCT_TIME_SECS	1444	X'000005A4'
V 9.3.0 V 9.3.0 MQIACF_LAST_UŻYWANE	1444	X'000005A4'

MQIACH_ * (typy kanałów całkowitoliczbowych w formacie komendy)

Tabela 186. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIACH_FIRST	1501	X'000005DD'
MQIACH_XMIT_PROTOCOL_TYPE	1501	X'000005DD'
MQIACH_BATCH_SIZE	1502	X'000005DE'

Tabela 186. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
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'

Tabela 186. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
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'

Tabela 186. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
MQIACH_AUTH_INFO_TYPES	1622	X'00000656'
MQIACH_RESET_REQUESTED	1623	X'00000657'
MQIACH_BATCH_DATA_LIMIT	1624	X'00000658'
MQIACH_MSG_HISTORY	1625	X'00000659'
MQIACH_MULTICAST_PROPERTIES	1626	X'0000065A'
MQIACH_NEW_SUBSCRIBER_HISTORY	1627	X'0000065B'
MQIACH_MC_HB_INTERVAL	1628	X'0000065C'
MQIACH_USE_CLIENT_ID	1629	X'0000065D'
MQIACH_MQTT_KEEP_ALIVE	1630	X'0000065E'
MQIACH_IN_DOUBT_IN	1631	X'0000065F'
MQIACH_IN_DOUBT_OUT	1632	X'00000660'
MQIACH_MSGS_SENT<	1633	X'00000661'
MQIACH_MSGS_RECEIVED	1634	X'00000662'

Tabela 186. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_PROTOCOL	1643	X'0000066B'
MQIACH_AMQPKEEPALIVE	1644	X'0000066C'
MQIACH_SECURITY_PROTOCOL	1645	X'0000066D'
z/OS MQIACH_SPL_PROTECTION	1646	X'0000066E'
MQIACH_LAST_USED	1646	X'0000066E'

MQIAMO_ * (Typy parametrów monitorowania liczb całkowitych w formacie komendy)

Tabela 187. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIAMO_FIRST	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'
KOMENDY MQIAMO_	710	X'000002C6'
MQIAMO_COMMITS_FAILED	711	X'000002C7'
MQIAMO_CONNS	712	X'000002C8'
MQIAMO_CONNS_MAX	713	X'000002C9'
Dyski MQIAMO_DISC	714	X'000002CA'
MQIAMO_DISCS_IMPLICIT,	715	X'000002CB'
MQIAMO_TYP_DYSKU	716	X'000002CC'
MQIAMO_CZAS_WYD. ŚREDNIA	717	X'000002CD'
MQIAMO_EXIT_TIME_MAX	718	X'000002CE'
CZAS WYJŚCIA MQIAMO_MIN	719	X'000002CF'
MQIAMO_FULL_PARTII	720	X'000002D0'
MQIAMO_GENERATED_MSGS	721	X'000002D1'

Tabela 187. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_GETS	722	X'000002D2'
MQIAMO_GET_MAX_BYTES	723	X'000002D3'
MQIAMO_GET_MIN_BYTES	724	X'000002D4'
Funkcja MQIAMO_GETS_FAILED	725	X'000002D5'
MQIAMO_INCOMPLETE_PARTII	726	X'000002D6'
MQIAMO_INQS	727	X'000002D7'
MQIAMO_MSGS	728	X'000002D8'
MQIAMO_CZAS_NET_ŚREDNIA	729	X'000002D9'
MQIAMO_NET_TIME_MAX	730	X'000002DA'
MQIAMO_NET_TIME_MIN	731	X'000002DB'
MQIAMO_OBJECT_COUNT	732	X'000002DC'
MQIAMO_OTWARTE	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_CZAS_Q_ŚREDNIA	741	X'000002E5'
MQIAMO_Q_TIME_MAX	742	X'000002E6'
MQIAMO_Q_TIME_MIN	743	X'000002E7'
MQIAMO_SETS	744	X'000002E8'
Funkcja MQIAMO_CONNS_FAILED	749	X'000002ED'
MQIAMO_OPENS_FAILED.	751	X'000002EF'
Niepowodzenie MQIAMO_INQS_FAILED	752	X'000002F0'
Komenda 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_XX_ENCODE_CASE_ONE komunikaty_nie_w_kolejce	759	X'000002F7'
MQIAMO_MSGS_PURGED	760	X'000002F8'
MQIAMO_SUBS_DUR	764	X'000002FC'
MQIAMO_SUBS_NDUR	765	X'000002FD'
Niepowodzenie MQIAMO_SUBS_FAILED	766	X'000002FE'
MQIAMO_SUBRQS	767	X'000002FF'
Niepowodzenie MQIAMO_SUBRQS_FAILED	768	X'00000300'
MQIAMO_CBS	769	X'00000301'

Tabela 187. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIAMO_CBS_FAILED,	770	X'00000302'
MQIAMO_CTLS	771	X'00000303'
Komenda MQIAMO_CTLS_FAILED	772	X'00000304'
MQIAMO_XX_ENCODE_CASE_ONE statystyki	773	X'00000305'
Niepowodzenie 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'
Komenda 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'
Komenda MQIAMO_UNSUBS_FAILED	788	X'00000314'
ODSTĘP_OKRES_MQIAM	789	X'00000315'
MQIAMO_MSGS_SENT	790	X'00000316'
MQIAMO_BYTES_SENT	791	X'00000317'
MQIAMO_NAPRAWY_BAJTY	792	X'00000318'
Tryb 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 (pakiety MQ)	799	X'0000031F'
SZYBKOŚĆ MQIAMO_PKT_RATE	800	X'00000320'
MQIAMO_MCAST_XMIT_RATE (szybkość przesyłania mcast_xmit_rate)	801	X'00000321'
MQIAMO_MCAST_BATCH_TIME	802	X'00000322'
MQIAMO_MCAST_HEARTBEAT	803	X'00000323'
MQIAMO_PORT_DANYCH_DOCELOWEGO	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'

Tabela 187. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIAMO_TOTAL_MSGS_SENT	811	X'0000032B'
MQIAMO_TOTAL_BYTES_SENT	812	X'0000032C'
MQIAMO_NUM_STREAMS	813	X'0000032D'
MQIAMO_ACK_FEEDBACK	814	X'0000032E'
MQIAMO_NACK_FEEDBACK	815	X'0000032F'
MQIAMO_PKTS_LOST	816	X'00000330'
MQIAMO_MSGS_RCVD	817	X'00000331'
MQIAMO_MSG_BYTES_RCVD	818	X'00000332'
MQIAMO_MSGS_DOSTARCZONE	819	X'00000333'
MQIAMO_PKTS_PROCESSED	820	X'00000334'
MQIAMO_PKTS_DLVD	821	X'00000335'
MQIAMO_PKTS_DROPPED	822	X'00000336'
MQIAMO_PKTS_ZDUPLIKOWANE	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 (łączna liczba wybranych komunikatów)	834	X'00000342'
MQIAMO_TOTAL_MSGS_EXPIRED:	835	X'00000343'
MQIAMO_TOTAL_MSGS_DOSTARCZONE	836	X'00000344'
MQIAMO_TOTAL_MSGS_ZWRÓCONE	837	X'00000345'
MQIAMO_LAST_USED	837	X'00000345'

MQIAMO64_* (64-bitowe typy parametrów monitorowania liczb całkowitych w formacie komendy)

Tabela 188. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'

Tabela 188. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (całkowitoliczbowe selektory systemowe)

Tabela 189. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIASY_FIRST	-1	X'FFFFFFFF'
MQIASY_CODED_CHAR_SET_ID	-1	X'FFFFFFFF'
TYP_MQIAS	-2	X'FFFFFFFE'
KOMENDA MQIASY_COMMAND	-3	X'FFFFFFFD'
MQIASY_MSG_SEQ_NUMBER	-4	X'FFFFFFFC'
MQIASY_CONTROL	-5	X'FFFFFFFB'
MQIASY_COMP_CODE	-6	X'FFFFFFFA'
MQIASY_REASON	-7	X'FFFFFFF9'
OPCJE MQIASY_BAG_OPCJE	-8	X'FFFFFFF8'
MQIASY_VERSION (wersja MQIAS)	-9	X'FFFFFFF7'
MQIASY_LAST_UŻYWANE	-9	X'FFFFFFF7'
MQIASY_LAST	-2000	X'FFFFF830'

MQIAUT_ * (element uwierzytelniający nagłówka informacji IMS)

Tabela 190. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQIAUT_BRAK	"¬¬¬¬¬¬¬¬"
MQIAUT_NONE_ARRAY	'¬','¬','¬','¬','¬','¬','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQIAV_ * (Wartości Atrybutów Całkowitych)

Tabela 191. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIAV_NOT_APPLICABLE (nie dotyczy)	-1	X'FFFFFFFF'
MQIAV_UNDEFINED,	-2	X'FFFFFFFE'

MQICM_ * (IMS informacje w nagłówku informacji o trybach zatwierdzania)

Tabela 192. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQICM_COMMIT_THEN_SEND	'0'
MQICM_SEND_THEN_COMMIT	'1'

MQIDO_ * (Opcje wątpliwe formatu komendy)

Tabela 193. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIDO_COMMIT	1	X'00000001'
WYCOFANIE MQIDO_BACKOUT	2	X'00000002'

MQIEP_ * (punkty wejścia interfejsu)

Struktura parametrów zabezpieczeń połączenia

Tabela 194. Struktury stałych	
Nazwa	**Struktura**
MQIEP_STRUC_ID (identyfikator struktury MQ)	"IEP¬"
MQIEP_STRUC_ID_ARRAY	'I','E','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 195. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIEP_VERSION_1	1	X'00000001'
MQDXP_CURRENT_VERSION	1	X'00000001'

MQIGQ_ * (kolejkowanie wewnątrz grupy)

Tabela 196. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIGQ_WYŁĄCZONE	0	X'00000000'
MQIGQ_ENABLED	1	X'00000001'

MQIGQPA_ * (uprawnienie do umieszczania w kolejce wewnątrz grupy)

Tabela 197. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIGQPA_DEFAULT	1	X'00000001'
MQIGQPA_CONTEXT	2	X'00000002'
MQIGQPA_ONLY_IGQ	3	X'00000003'
MQIGQPA_ALTERNATE_LUB_IGQ	4	X'00000004'

MQIIH_ * (struktura i flagi nagłówka informacji IMS)

Struktura nagłówka informacji IMS

Tabela 198. Struktury stałych	
Nazwa	**Struktura**
MQIIH_ID_struktury	"IIH¬"
MQIIH_STRUC_ID_ARRAY	'I','I','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 199. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIIH_VERSION_1	1	X'00000001'
MQIIH_CURRENT_VERSION	1	X'00000001'
MQIIH_LENGTH_1	84	X'00000054'

IMS flagi nagłówka informacji

Tabela 200. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIIH_BRAK	0	X'00000000'
MQIIH_PASS_EXPIRATION	1	X'00000001'
MQIIH_UNLI_XD_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_ * (zapytanie o opcje i strukturę właściwości komunikatu)

Sprawdź strukturę opcji właściwości komunikatu

Tabela 201. Struktury stałych	
Nazwa	**Struktura**
MQIMPO_ID_struktury	"IMPO"
MQIMPO_STRUC_ID_ARRAY	'I','M','P','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 202. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIMPO_VERSION_1	1	X'00000001'
MQIMPO_CURRENT_VERSION	1	X'00000001'

Sprawdź opcje właściwości komunikatu

Tabela 203. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQIMPO_CONVERT_TYPE	2	X'00000002'
DŁUGOŚĆ KOLEJKI MQIMPO_QUERY_LENGTH	4	X'00000004'
MQIMPO_INQ_FIRST	0	X'00000000'
MQIMPO_INQ_NEXT	8	X'00000008'
MQIMPO_INQ_PROP_UNDER_CURSOR	16	X'00000010'
MQIMPO_KONVERT_VALUE	32	X'00000020'
MQIMPO_BRAK	0	X'00000000'

MQINBD_ * (Rozdysponowanie danych przychodzących w formacie komendy)

Tabela 204. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQINBD_Q_MGR	0	X'00000000'
GRUPA MQINBD	3	X'00000003'

MQIND_ * (specjalne wartości indeksu)

Tabela 205. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIND_BRAK	-1	X'FFFFFFFF'
MQIND_ALL (wszystkie)	-2	X'FFFFFFFE'

MQIPADDR_ * (wersje adresu IP)

Tabela 206. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIPADDR_IPV4	0	X'00000000'
MQIPADDR_IPV6	1	X'00000001'

MQISS_ * (zasięgi zabezpieczeń nagłówka informacji IMS)

Tabela 207. Nazwy i wartości stałych

Nazwa	Wartość
MQISS_CHECK	'C'
MQISS_FULL	'F'

MQIT_ * (typy indeksów)

Tabela 208. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQIT_BRAK	0	X'00000000'
MQIT_MSG_ID	1	X'00000001'
MQIT_CORREL_ID (Identyfikator KORELACJI MQ)	2	X'00000002'
MQIT_MSG_TOKEN,	4	X'00000004'
MQIT_GROUP_ID (identyfikator grupy MQT)	5	X'00000005'

MQITEM_ * (typ elementu dla komendy mqInquireItemInfo)

Tabela 209. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQITEM_INTEGER	1	X'00000001'
ŁAŃCUCH_ARTYKULU MQITEM	2	X'00000002'
MQITEM_BAG	3	X'00000003'
MQITEM_BYTE_STRING ŁAŃCUCH	4	X'00000004'
FILTR LICZBY CAŁKOWITEJ MQITEM_INTEGER_FILTER	5	X'00000005'

Tabela 209. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
FILTR ŁAŃCUCHÓW MQITEM_STRING_FILTER	6	X'00000006'
MQITEM_INTEGER64	7	X'00000007'
FILTR MQITEM_BYTE_STRING_FILTER	8	X'00000008'

MQITII_ * (identyfikator instancji transakcji nagłówka informacji IMS)

Tabela 210. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQITII_BRAK	X'00...00' (16 wartości pustych)
MQITII_NONE_ARRAY	'\0','\0',... (16 wartości pustych)

MQITS_ * (stany transakcji nagłówka informacji IMS)

Tabela 211. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQITS_IN_CONVERSATION	'C'
MQITS_NOT_IN_CONVERSACJA	'¬'
MQITS_ARCHITECTED	'A'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQKAI_ * (przedział czasuKeepAlive)

Tabela 212. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQKAI_AUTO	-1	X'FFFFFFFF'

MQMASTER_ * (Administrowanie główne)

Tabela 213. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMASTER_NO	0	X'00000000'
MQMASTER_YES	1	X'00000001'

MQMCAS_ * (Status agenta kanału komunikatów w formacie komendy)

Tabela 214. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMCAS_ZATRZYMANY	0	X'00000000'
MQMCAS_RUNNING	3	X'00000003'

MQMCAT_ * (typy MCA)

Tabela 215. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
PROCES_MQMCAT	1	X'00000001'

Tabela 215. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
WĄTEK MQMCAT_THREAD	2	X'00000002'

MQMCD_ * (informacje o znaczniku opcji publikowania/subskrypcji)

Znaczniki deskryptora treści komunikatu (mcd) znacznika opcji publikowania/subskrypcji

Tabela 216. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
WERSJA_FOLDERU_MQMC	1	X'00000001'

Nazwy znaczników opcji publikowania/subskrypcji

Tabela 217. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQMCD_MSG_DOMAIN,	"Msd"
MQMCD_MSG_SET	"Set"
TYP_KOMUNIKATU_MQMCD	"Type"
MQMCD_MSG_FORMAT	"Fmt"

Nazwy znaczników XML opcji publikowania/subskrypcji

Tabela 218. Nazwy i wartości stałych	
Nazwa	**Wartość**
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>"
MQMCD_MSG_TYPE_E	"</Type>"
MQMCD_MSG_FORMAT_B	"<Fmt>"
MQMCD_MSG_FORMAT_E	"</Fmt>"

Wartości znaczników opcji publikowania/subskrypcji

Tabela 219. Nazwy i wartości stałych	
Nazwa	**Wartość**
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"

Tabela 219. Nazwy i wartości stałych (kontynuacja)	
Nazwa	**Wartość**
MQMCD_DOMAIN_JMS_STREAM	"jms_stream"
MQMCD_DOMAIN_JMS_BYTES,	"jms_bytes"

MQMD_ * (struktura deskryptora komunikatu)

Tabela 220. Struktury stałych	
Nazwa	**Struktura**
MQMD_STRUC_ID (Identyfikator struktury kolejki)	"MD¬¬"
MQMD_STRUC_ID_ARRAY	'M','D','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 221. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMD_VERSION_1	1	X'00000001'
MQMD_VERSION_2	2	X'00000002'
MQMD_CURRENT_VERSION	2	X'00000002'

MQMDE_ * (struktura rozszerzenia deskryptora komunikatu)

Tabela 222. Struktury stałych	
Nazwa	**Struktura**
MQMDE_STRUC_ID (Identyfikator struktury menedżera kolejek)	"MDE¬"
MQMDE_STRUC_ID_ARRAY	'M','D','E','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 223. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMDE_VERSION_2	2	X'00000002'
MQMDE_CURRENT_VERSION	2	X'00000002'
MQMDE_LENGTH_2	72	X'00000048'

MQMDEF_ * (Flagi rozszerzenia deskryptora komunikatu)

Tabela 224. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMDEF_BRAK	0	X'00000000'

MQMDS_ * (kolejność dostarczania komunikatów)

Tabela 225. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMDS_PRIORITY,	0	X'00000000'
MQMDS_FIFO,	1	X'00000001'

MQMF_ * (flagi komunikatu)

Tabela 226. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMF_SEGMENTATION_INHIBITED	0	X'00000000'
MQMF_SEGMENTATION_ALLOWED	1	X'00000001'
MQMF_MSG_IN_GROUP	8	X'00000008'
MQMF_LAST_MSG_IN_GROUP	16	X'00000010'
MQMF_SEGMENT	2	X'00000002'
MQMF_LAST_SEGMENT	4	X'00000004'
MQMF_BRAK	0	X'00000000'

MQMHBO_ * (uchwyt komunikatu do opcji buforu i struktury)

Struktura opcji przesyłania komunikatu do buforu

Tabela 227. Struktury stałych	
Nazwa	**Struktura**
MQMHBO_STRUC_ID (Identyfikator struktury menedżera kolejek)	"MHBO"
MQMHBO_STRUC_ID_ARRAY	'M','H','B','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 228. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMHBO_VERSION_1	1	X'00000001'
MQMHBO_CURRENT_VERSION	1	X'00000001'

Opcje uchwytu komunikatu do buforu

Tabela 229. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMHBO_PROPERTIES_IN_MQRFH2	1	X'00000001'
MQMHBO_DELETE_PROPERTIES	2	X'00000002'
MQMHBO_BRAK	0	X'00000000'

MQMI_ * (identyfikator komunikatu)

Tabela 230. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQMI_BRAK	X'00...00' (24 wartości null)
MQMI_NONE_ARRAY	'\0','\0',... (24 wartości null)

MQMMBI_ * (Znacznik komunikatu-odstęp czasu przeglądania)

Tabela 231. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMMBI_XX_ENCODE_CASE_ONE nieograniczone	-1	X'FFFFFFFF'

MQMO_ * (opcje dopasowania)

Tabela 232. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMO_MATCH_MSG_ID	1	X'00000001'
MQMO_MATCH_CORREL_ID	2	X'00000002'
MQMO_MATCH_GROUP_ID (identyfikator grupy zgodności MQ)	4	X'00000004'
MQMO_MATCH_MSG_SEQ_NUMBER	8	X'00000008'
MQMO_ZGODNE_PRZESUNIĘCIE	16	X'00000010'
MQMO_MATCH_MSG_TOKEN,	32	X'00000020'
MQMO_BRAK	0	X'00000000'

MQMODE_ * (Opcje trybu formatu komendy)

Tabela 233. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMODE_FORCE	0	X'00000000'
MQMODE_QUIESCE,	1	X'00000001'
MQMODE_TERMINATE	2	X'00000002'

MQMON_ * (wartości monitorowania)

Tabela 234. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMON_NIEDOSTĘPNE	-1	X'FFFFFFFF'
MQMON_BRAK	-1	X'FFFFFFFF'
MQMON_Q_MGR	-3	X'FFFFFFFD'
MQMON_WYŁ	0	X'00000000'
MQMON_ON	1	X'00000001'
MQMON_DISABLED (wyłączone)	0	X'00000000'
MQMON_ENABLED	1	X'00000001'
MQMON_NISKI	17	X'00000011'
MQMON_MEDIUM	33	X'00000021'
MQMON_HIGH	65	X'00000041'

MQMT_ * (typy komunikatów)

Tabela 235. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQMT_SYSTEM_FIRST	1	X'00000001'
MQMT_ŻĄDANIE	1	X'00000001'
MQMT_REPLY	2	X'00000002'
MQMT_DATAGRAM,	8	X'00000008'
MQMT_RAPORT	4	X'00000004'
MQMT_MQE_FIELDS_FROM_MQE	112	X'00000070'

Tabela 235. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQMT_MQE_FIELDS	113	X'00000071'
MQMT_SYSTEM_LAST	65535	X'0000FFFF'
MQMT_APPL_FIRST	65536	X'00010000'
MQMT_APPL_LAST	999999999	X'3B9AC9FF'

MQMTOK_ * (znacznik komunikatu)

Tabela 236. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQMTOK_BRAK	X'00...00' (16 wartości pustych)
MQMTOK_NONE_ARRAY	'\0','\0',... (16 wartości pustych)

MQNC_ * (Liczba nazw)

Tabela 237. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQNC_MAX_NAMELIST_NAME_COUNT,	256	X'00000100'

MQNPM_ * (klasa komunikatu nietrwałego)

Tabela 238. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQNPM_CLASS_NORMAL	0	X'00000000'
MQNPM_CLASS_HIGH,	10	X'0000000A'

MQNPMS_ * (NonPersistent-Szybkość komunikatów)

Tabela 239. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQNPMS_NORMAL	1	X'00000001'
MQNPMS_FAST	2	X'00000002'

MQNT_ * (Typy list nazw)

Tabela 240. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (Nazwy dla łańcucha nazwa/wartość)

Tabela 241. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQNVS_APPL_TYPE	"OPT_APP_GRP¬"
MQNVS_MSG_TYPE	"OPT_MSG_TYPE¬"

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQOA_ * (limity dla selektorów dla atrybutów obiektu)

Tabela 242. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOA_FIRST	1	X'00000001'
MQOA_LAST	9000	X'00002328'

MQOD_ * (struktura deskryptora obiektu)

Tabela 243. Struktury stałych	
Nazwa	**Struktura**
MQOD_STRUC_ID (identyfikator struktury MQD)	"OD¬¬"
MQOD_STRUC_ID_ARRAY	'O','D','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 244. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOD_VERSION_1	1	X'00000001'
MQOD_VERSION_2	2	X'00000002'
MQOD_VERSION_3	3	X'00000003'
MQOD_VERSION_4	4	X'00000004'
MQOD_CURRENT_VERSION	4	X'00000004'
MQOD_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

MQOII_ * (identyfikator instancji obiektu)

Tabela 245. Nazwy i wartości stałych	
Nazwa	**Wartość**
MQOII_BRAK	X'00...00' (24 wartości null)
MQOII_NONE_ARRAY	'\0','\0',... (24 wartości null)

MQOL_ * (pierwotna długość)

Tabela 246. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOL_UNDEFINED	-1	X'FFFFFFFF'

MQOM_ * (przestarzałe opcje komunikatów Db2 w grupie uzyskiwania informacji)

Tabela 247. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOM_NO	0	X'00000000'
MQOM_TAK	1	X'00000001'

MQOO_ * (opcje otwarcia)

Tabela 248. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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 (wyłączna)	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 (uprawnienie użytkownika na przemian)	4096	X'00001000'
MQOO_FAIL_IF_QUIESCING,	8192	X'00002000'
MQOO_BIND_ON_OPEN-OTWARTE	16384	X'00004000'
MQOO_BIND_NOT_FIXED (NIEUSTALONY)	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_ * (używane tylko w języku C++)*

Tabela 249. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQOO_RESOLVE_NAMES	65536	X'00010000'
MQOO_RESOLVE_LOCAL_Q	262144	X'00040000'

MQOP_ * (kody operacji dla MQCTL i MQCB)

Kody operacji dla MQCTL

Tabela 250. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOP_START	1	X'00000001'
OCZEKIWANIE_NA_URUCHOM_MQOPP	2	X'00000002'
MQOP_STOP	4	X'00000004'

Kody operacji dla obiektu MQCB

Tabela 251. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOP_REGISTER,	256	X'00000100'
MQOP_DEREGISTER	512	X'00000200'

Kody operacji dla MQCTL i MQCB

Tabela 252. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOP_ZAWIEŚ	65536	X'00010000'
MQOP_RESUME	131072	X'00020000'

MQOPEN_ * (wartości związane ze strukturą MQOPEN_PRIV)

Tabela 253. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOPEN_PRIV_VERSION_1	1	X'00000001'
MQOPEN_PRIV_CURRENT_VERSION	1	X'00000001'

MQOPER_ * (operacje działania)

Tabela 254. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOPER_SYSTEM_FIRST	0	X'00000000'
MQOPER_UNKNOWN	0	X'00000000'
MQOPER_BROWSE,	1	X'00000001'
MQOPER_DISCARD,	2	X'00000002'
MQOPER_GET	3	X'00000003'
MQOPER_PUT	4	X'00000004'
MQOPER_PUT_REPLY,	5	X'00000005'
RAPORT MQOPER_PUT_REPORT	6	X'00000006'
MQOPER_RECEIVE	7	X'00000007'
MQOPER_SEND	8	X'00000008'
MQOPER_TRANSFORM	9	X'00000009'

Tabela 254. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOPER_PUBLISH	10	X'0000000A'
MQOPER_EXCLUDED_PUBLISH	11	X'0000000B'
MQOPER_DISCARDED_PUBLISH	12	X'0000000C'
MQOPER_SYSTEM_LAST	65535	X'0000FFFF'
MQOPER_APPL_FIRST	65536	X'00010000'
MQOPER_APPL_LAST	999999999	X'3B9AC9FF'

MQOT_ * (typy obiektów i rozszerzone typy obiektów)

Typy obiektów

Tabela 255. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOT_BRAK	0	X'00000000'
MQOT_Q	1	X'00000001'
LISTA NAZW MQOT_NAMELIST	2	X'00000002'
PROCES_MQOT	3	X'00000003'
KLASA MQOT_STORAGE_CLASS	4	X'00000004'
MQOT_Q_MGR	5	X'00000005'
MQOT_CHANNEL (kanał MQT)	6	X'00000006'
MQOT_AUTH_INFO	7	X'00000007'
TEMAT_MQOT	8	X'00000008'
MQOT_CF_STRUC	10	X'0000000A'
MQOT_LISTENER	11	X'0000000B'
USŁUACJA_MQOT	12	X'0000000C'
MQOT_RESERVED_1	999	X'000003E7'

Rozszerzone typy obiektów

Tabela 256. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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 (kanał nadawczy)	1007	X'000003EF'
MQOT_SERVER_CHANNEL,	1008	X'000003F0'
MQOT_REQUESTER_CHANNEL	1009	X'000003F1'
MQOT_RECEIVER_CHANNEL,	1010	X'000003F2'
MQOT_BIEŻĄCY_KANAŁ	1011	X'000003F3'
MQOT_SAVED_CHANNEL (kanał zapisu MQ)	1012	X'000003F4'

Tabela 256. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQOT_SVRCONN_CHANNEL, KANAŁ	1013	X'000003F5'
MQOT_CLNTCONN_CHANNEL, kanał	1014	X'000003F6'
MQOT_SHORT_CHANNEL	1015	X'000003F7'
MQOT_CHLAUTH	1016	X'000003F8'
MQOT_REMOTE_Q_MGR_NAME	1017	X'000003F9'
MQOT_PROT_POLICY	1019	X'000003FB'
MQOT_TT_CHANNEL,	1020	X'000003FC'
MQOT_AMQP_CHANNEL	1021	X'000003FD'
MQOT_AUTH_REC	1022	X'000003FE'

MQPA_ * (uprawnienie do umieszczania)

Tabela 257. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPA_DEFAULT	1	X'00000001'
MQPA_CONTEXT	2	X'00000002'
MQPA_ONLY_MCA	3	X'00000003'
MQPA_ALTERNATE_OR_MCA	4	X'00000004'

MQPD_ * (deskryptor właściwości, obsługa i kontekst)

Struktura deskryptora właściwości

Tabela 258. Struktury stałych	
Nazwa	**Struktura**
MQPD_STRUC_ID	"PD¬¬"
MQPD_STRUKTURA_ID_ARRAY	'P','D','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 259. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPD_VERSION_1	1	X'00000001'
MQPD_CURRENT_VERSION	1	X'00000001'

Opcje deskryptora właściwości

Tabela 260. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPD_BRAK	0	X'00000000'

Opcje obsługi właściwości

Tabela 261. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Kontekst właściwości

Tabela 262. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPD_NO_CONTEXT	0	X'00000000'
MQPD_USER_CONTEXT	1	X'00000001'

MQPER_ * (wartości trwałości)

Tabela 263. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (platformy)

Tabela 264. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'
Maszyny wirtualne MQPL_VMS	12	X'0000000C'
MQPL_NSK	13	X'0000000D'
MQPL_OPEN_TP1	15	X'0000000F'
Maszyna wirtualna MQPL_VM	18	X'00000012'

Tabela 264. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPL_TPF	23	X'00000017'
MQPL_VSE	27	X'0000001B'
MQPL_APPLIANCE	28	X'0000001C'
RODZIMA MQPL_NATIVE	1	X'00000001'

MQPMO_ * (Opcje umieszczania komunikatów i struktura dla maski publikowania)

Struktura opcji umieszczania komunikatu

Tabela 265. Struktury stałych	
Nazwa	**Struktura**
MQPMO_STRUC_ID,	"PMO¬"
MQPMO_STRUC_ID_ARRAY	'P','M','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 266. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_CURRENT_LENGTH	(value differs by platform or version)	(value differs by platform or version)

Opcje umieszczania komunikatu

Tabela 267. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_UPRAWNIENIA_UŻYTKOWNIKA	4096	X'00001000'
MQPMO_FAIL_IF_QUIESCING,	8192	X'00002000'
MQPMO_NO_CONTEXT	16384	X'00004000'
MQPMO_LOGICAL_ORDER,	32768	X'00008000'

Tabela 267. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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 (menedżer kolejek)	67108864	X'04000000'
MQPMO_SUPPRESS_REPLYTO	134217728	X'08000000'
MQPMO_NOT_OWN_SUBS,	268435456	X'10000000'
MQPMO_ODPOWIEDŹ_AS_Q_DEF	0	X'00000000'
MQPMO_ODPOWIEDŹ_AS_TOPIC_DEF	0	X'00000000'
MQPMO_BRAK	0	X'00000000'

Opcje umieszczania komunikatu dla maski publikowania

Tabela 268. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPMO_PUB_OPTIONS_MASK	2097152	X'00200000'

MQPMRF_ * (pola rekordu umieszczania komunikatu)

Tabela 269. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPMRF_ID_komunikatu	1	X'00000001'
MQPMRF_CORREL_ID (Identyfikator relacji MQ)	2	X'00000002'
MQPMRF_GROUP_ID (identyfikator grupy MQPMRF_ID)	4	X'00000004'
MQPMRF_FEEDBACK	8	X'00000008'
MQPMRF_ACCOUNTING_TOKEN,	16	X'00000010'
MQPMRF_BRAK	0	X'00000000'

MQPO_ * (opcje czyszczenia formatu komendy)

Tabela 270. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPO_TAK	1	X'00000001'
MQPO_NIE	0	X'00000000'

MQPRI_ * (priorytet)

Tabela 271. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPRI_PRIORITY_AS_Q_DEF	-1	X'FFFFFFFF'
MQPRI_PRIORITY_AS_PARENT	-2	X'FFFFFFFE'
MQPRI_PRIORITY_AS_PUBLISHED	-3	X'FFFFFFFD'

Tabela 271. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPRI_PRIORITY_AS_TOPIC_DEF	-1	X'FFFFFFFF'

MQPROP_ * (wartości kontrolne właściwości kolejki i kanału oraz maksymalna długość właściwości)

Wartości kontrolne właściwości kolejki i kanału

Tabela 272. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPROP_KOMPATYBILNOŚĆ	0	X'00000000'
MQPROP_NONE	1	X'00000001'
MQPROP_ALL	2	X'00000002'
MQPROP_FORCE_MQRFH2	3	X'00000003'

Maksymalna długość właściwości

Tabela 273. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPROP_UNRESTRICTED_LENGTH	-1	X'FFFFFFFF'

MQPRT_ * (Wartości odpowiedzi umieszczania)

Tabela 274. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPRT_RESPONSE_AS_PARENT	0	X'00000000'
ODPOWIEDŹ MQPRT_SYNC_RESPONSE	1	X'00000001'
MQPRT_ASYNC_RESPONSE	2	X'00000002'

MQPS_ * (publikowanie/subskrybowanie)

Status publikowania/subskrybowania w formacie komendy

Tabela 275. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQPS_STATUS_INACTIVE (aktywny)	0	X'00000000'
MQPS_STATUS_STARTING	1	X'00000001'
MQPS_STATUS_ZATRZYMANY	2	X'00000002'
MQPS_STATUS_ACTIVE	3	X'00000003'
MQPS_STATUS_COMPAT	4	X'00000004'
MQPS_STATUS_ERROR	5	X'00000005'
MQPS_STATUS_REFUSED	6	X'00000006'

Publikuj/subskrybuj znaczniki jako łańcuchy

MQPS_COMMAND	"MQPSCommand"

KOD_COMP_MQPS_COMP_CODE	"MQPSCompCode"
MQPS_CORREL_ID	"MQPSCorrelId"
OPCJE MQPS_DELETE_OPTIONS	"MQPSDelOpts"
MQPS_ERROR_ID	"MQPSErrorId"
MQPS_ERROR_POS	"MQPSErrorPos"
DANE PRODUKTU MQPS_INTEGER_DATA	"MQPSIntData"
ID_PARAMETER_MQPS	"MQPSParmId"
OPCJE PRODUKTU MQPS_PUBLIC	"MQPSPubOpts"
ZNACZNIK CZASU PUBLIKOWANIA MQPS_PUBLISH_TIMESTAMP	"MQPSPubTime"
MQPS_Q_MGR_NAME	"MQPSQMgrName"
MQPS_Q_NAME	"MQPSQName"
Produkt MQPS_REASON	"MQPSReason"
MQPS_REASON_TEXT	"MQPSReasonText"
MQPS_REGISTRATION_OPTIONS (Opcje rejestracji MQPS)	"MQPSRegOpts"
NUMER PRODUKTU MQPS_SEQUENCE_NUMBER	"MQPSSeqNum"
MQPS_STREAM_NAME	"MQPSStreamName"
DANE PRODUKTU MQPS_STRING_DATA	"MQPSStringData"
MQPS_SUBSCRIPTION_IDENTITY	"MQPSSubIdentity"
MQPS_SUBSCRIPTION_NAME	"MQPSSubName"
MQPS_SUBSCRIPTION_USER_DATA	"MQPSSubUserData"
MQPS_TOPIC	"MQPSTopic"
ID_UŻYTKOWNIKA MQPS_USER_ID	"MQPSUserId"

Publikuj/subskrybuj znaczniki w postaci pustych łańcuchów

MQPS_COMMAND_B	"¬MQPSCommand¬"
MQPS_COMP_CODE_B	"¬MQPSCompCode¬"
MQPS_CORREL_ID_B	"¬MQPSCorrelId¬"
Produkt MQPS_DELETE_OPTIONS_B	"¬MQPSDelOpts¬"
ID_BŁĘDU_MQB	"¬MQPSErrorId¬"
MQPS_ERROR_POS_B	"¬MQPSErrorPos¬"
MQPS_INTEGER_DATA_B	"¬MQPSIntData¬"
ID_PARAMETER_B MQPS_PARAMETER_B	"¬MQPSParmId¬"
MQPS_PUBLIC OPTIONS_B	"¬MQPSPubOpts¬"
MQPS_PUBLISH_TIMESTAMP_B	"¬MQPSPubTime¬"
MQPS_Q_MGR_NAME_B	"¬MQPSQMgrName¬"
MQPS_Q_NAME_B	"¬MQPSQName¬"

MQPS_REASON_B	"¬MQPSReason¬"
MQPS_REASON_TEXT_B	"¬MQPSReasonText¬"
MQPS_REGISTRATION_OPTIONS_B	"¬MQPSRegOpts¬"
MQPS_SEQUENCE_NUMBER_B	"¬MQPSSeqNum¬"
MQPS_STREAM_NAME_B	"¬MQPSStreamName¬"
MQPS_STRING_DATA_B	"¬MQPSStringData¬"
MQPS_SUBSCRIPTION_IDENTITY_B	"¬MQPSSubIdentity¬"
MQPS_SUBSCRIPTION_NAME_B	"¬MQPSSubName¬"
MQPS_SUBSCRIPTION_USER_DATA_B	"¬MQPSSubUserData¬"
MQPS_TOPIC_B	"¬MQPSTopic¬"
ID_UŻYTKOWNIKA MQPS_B	"¬MQPSUserId¬"

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Wartości znaczników komend publikowania/subskrypcji jako łańcuchy

PUBLIKACJA MQPS_DELETE_PUBLACJI	"DeletePub"
MQPS_DEREGISTER_PUBLISHER	"DeregPub"
MQPS_DEREGISTER_SUBSCRIBER	"DeregSub"
MQPS_PUBLISH	"Publish"
MQPS_REGISTER_PUBLISHER	"RegPub"
MQPS_REGISTER_SUBSCRIBER	"RegSub"
MQPS_REQUEST_UPDATE	"ReqUpdate"

Wartości znaczników komend publikowania/subskrypcji w postaci pustych łańcuchów

MQPS_DELETE_PUBLIC B	"¬DeletePub¬"
MQPS_DEREGISTER_PUBLISHER_B	"¬DeregPub¬"
MQPS_DEREGISTER_SUBSCRIBER_B	"¬DeregSub¬"
MQPS_PUBLISH_B	"¬Publish¬"
MQPS_REGISTER_PUBLISHER_B	"¬RegPub¬"
MQPS_REGISTER_SUBSCRIBER_B	"¬RegSub¬"
MQPS_REQUEST_UPDATE_B	"¬ReqUpdate¬"

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Wartości znaczników opcji publikowania/subskrypcji jako łańcuchy

MQPS_ADD_NAME	"AddName"
MQPS_ANONYMOUS	"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_ZACHOWANY	"InformIfRet"
MQPS_IS_RETAINED_PUBLIKACJA	"IsRetainedPub"
NIEROZSTRZYGNIĘTE MQPS_JOIN_	"JoinExcl"
MQPS_JOIN_SHARED	"JoinShared"
TYLKO PRODUKT MQPS_LEAVE_ONLY	"LeaveOnly"
MQPS_LOCAL	"Local"
MQPS_LOCKED	"Locked"
MQPS_NEW_PUBLIC TYLKO	"NewPubsOnly"
MQPS_NO_ALTERATION	"NoAlter"
MQPS_NO_REGISTRATION	"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_RETAIN_PUBLIKACJA	"RetainPub"
MQPS_VARIABLE_USER_ID	"VariableUserId"

Wartości znaczników opcji publikowania/subskrypcji w postaci pustych łańcuchów

MQPS_ADD_NAME_B	"¬AddName¬"
MQPS_ANONYMOUS_B	"¬Anon¬"
MQPS_CORREL_ID_AS_IDENTITY_B	"¬CorrelAsId¬"
MQPS_DEREGISTER_ALL_B	"¬DeregAll¬"
MQPS_DIRECT_REQUESTS_B	"¬DirectReq¬"
MQPS_DUPLICATES_OK_B	"¬DupsOK¬"
MQPS_FULL_RESPONSE_B	"¬FullResp¬"
MQPS_INCLUDE_STREAM_NAME_B	"¬InclStreamName¬"
MQPS_INFORM_IF_RETAINED_B	"¬InformIfRet¬"
MQPS_IS_RETAINED_PUBLIC B	"¬IsRetainedPub¬"

MQPS_JOIN_EXCLUSIVE_B	"¬JoinExcl¬"
MQPS_JOIN_SHARED_B	"¬JoinShared¬"
MQPS_LEAVE_ONLY_B	"¬LeaveOnly¬"
MQPS_LOCAL_B	"¬Local¬"
MQPS_LOCKED_B	"¬Locked¬"
MQPS_NEW_PUBLIC ONLY_B	"¬NewPubsOnly¬"
MQPS_NO_ALTERATION_B	"¬NoAlter¬"
MQPS_NO_REGISTRATION_B	"¬NoReg¬"
MQPS_NON_PERSISTENT_B	"¬NonPers¬"
MQPS_NONE_B	"¬None¬"
MQPS_OTHER_SUBSCRIBERS_ONLY_B	"¬OtherSubsOnly¬"
Tabela MQPS_PERSISTENT_B	"¬Pers¬"
MQPS_PERSISTENT_AS_PUBLISH_B	"¬PersAsPub¬"
MQPS_PERSISTENT_AS_Q_B	"¬PersAsQueue¬"
MQPS_PUBLISH_ON_REQUEST_ONLY_B	"¬PubOnReqOnly¬"
MQPS_RETAIN_PUBLIC B	"¬RetainPub¬"
MQPS_VARIABLE_USER_ID_B	"¬VariableUserId¬"

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQPSC_ * (znaczniki folderu komend publikowania/subskrypcji (psc) znacznika opcji publikowania/subskrypcji)

Tabela 276. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSC_FOLDER_VERSION,	1	X'00000001'

*MQPSC_ * (nazwy znaczników opcji publikowania/subskrypcji)*

KOMENDA MQPSC_COMMAND	"Command"
Opcja MQPSC_REGISTRATION_OPTION	"RegOpt"
OPCJA MQPSC_PUBLIC	"PubOpt"
MQPSC_DELETE_OPCJA	"DelOpt"
MQPSC_TOPIC	"Topic"
MQPSC_SUBSCRIPTION_POINT	"SubPoint"
MQPSC_FILTR	"Filter"
MQPSC_Q_MGR_NAME	"QMgrName"
MQPSC_Q_NAME	"QName"
MQPSC_PUBLISH_TIMESTAMP	"PubTime"
MQPSC_SEQUENCE_NUMBER	"SeqNum"

MQPSC_SUBSCRIPTION_NAME	"SubName"
MQPSC_SUBSCRIPTION_IDENTITY,	"SubIdentity"
MQPSC_SUBSCRIPTION_USER_DATA	"SubUserData"
MQPSC_CORREL_ID,	"CorrelId"

*MQPSC_ * (nazwy znaczników XML opcji publikowania/subskrypcji)*

MQPSC_COMMAND_B	"<Command>"
MQPSC_COMMAND_E,	"</Command>"
MQPSC_REGISTRATION_OPTION_B	"<RegOpt>"
MQPSC_REGISTRATION_OPTION_E	"</RegOpt>"
MQPSC_PUBLIC OPTION_B	"<PubOpt>"
MQPSC_PUBLIC OPTION_E	"</PubOpt>"
MQPSC_DELETE_OPTION_B	"<DelOpt>"
MQPSC_DELETE_OPTION_E,	"</DelOpt>"
MQPSC_TOPIC_B	"<Topic>"
MQPSC_TOPIC_E	"</Topic>"
Punkt subskrypcji MQPSC_SUBSCRIPTION_POINT_B	"<SubPoint>"
MQPSC_PUNKT_SUBSKRYPCJI e	"</SubPoint>"
MQPSC_FILTER_B	"<Filter>"
MQPSC_FILTER_E	"</Filter>"
MQPSC_Q_MGR_NAME_B	"<QMgrName>"
MQPSC_Q_MGR_NAME_E	"</QMgrName>"
MQPSC_Q_NAME_B	"<QName>"
MQPSC_Q_NAME_E	"</QName>"
MQPSC_PUBLISH_TIMESTAMP_B	"<PubTime>"
MQPSC_PUBLISH_TIMESTAMP_E	"</PubTime>"
MQPSC_SEQUENCE_NUMBER_B	"<SeqNum>"
MQPSC_SEQUENCE_NUMBER_E	"</SeqNum>"
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>"
MQPSC_CORREL_ID_E	"</CorrelId>"

*MQPSC_ * (wartości publikowania/subskrybowania znaczników publikatora jako łańcuchy)*

MQPSC_DELETE_PUBLIKACJA	"DeletePub"
MQPSC_DEREGISTER_SUBSCRIBER	"DeregSub"
MQPSC_PUBLISH	"Publish"
MQPSC_REGISTER_SUBSCRIBER	"RegSub"
MQPSC_REQUEST_UPDATE	"ReqUpdate"

*MQPSC_ * (wartości nazw znaczników opcji publikowania/subskrypcji jako łańcuchy)*

MQPSC_ADD_NAME	"AddName"
MQPSC_CORREL_ID_AS_IDENTITY	"CorrelAsId"
MQPSC_DEREGISTER_ALL	"DeregAll"
MQPSC_DUPLICATES_OK	"DupsOK"
MQPSC_FULL_RESPONSE	"FullResp"
MQPSC_INFORM_IF_ZACHOWANY	"InformIfRet"
MQPSC_IS_RETAINED_PUB	"IsRetainedPub"
MQPSC_JOIN_SHARED	"JoinShared"
MQPSC_JOIN_EXCLUSIVE	"JoinExcl"
MQPSC_LEAVE_ONLY,	"LeaveOnly"
MQPSC_LOCAL,	"Local"
MQPSC_ZABLOKOWANY	"Locked"
MQPSC_NEW_PUBS_ONLY,	"NewPubsOnly"
MQPSC_NO_ALTERATION	"NoAlter"
MQPSC_NON_PERSISTENT	"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_XX_ENCODE_CASE_ONE zmienna_id_użytkownika	"VariableUserId"

MQPSCR_ * (opcje publikowania/subskrypcji)

Znaczniki folderu odpowiedzi publikowania/subskrypcji (pscr) znacznika opcji publikowania/subskrypcji

Tabela 277. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSCR_FOLDER_VERSION,	1	X'00000001'

Nazwy znaczników opcji publikowania/subskrypcji

MQPSCR_COMPLETION,	"Completion"
MQPSCR_RESPONSE	"Response"
MQPSCR_REASON.	"Reason"

Nazwy znaczników XML opcji publikowania/subskrypcji

MQPSCR_COMPLETION_B	"<Completion>"
MQPSCR_COMPLETION_E	"</Completion>"
MQPSCR_RESPONSE_B	"<Response>"
MQPSCR_RESPONSE_E	"</Response>"
MQPSCR_REASON_B	"<Reason>"
MQPSCR_REASON_E	"</Reason>"

Wartości znaczników opcji publikowania/subskrypcji

MQPSCR_OK	"ok"
MQPSCR_OSTRZEŻENIE	"warning"
MQPSCR_BŁĄD	"error"

MQPSM_ * (tryb publikowania/subskrypcji)

Tabela 278. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSM_WYŁĄCZONE	0	X'00000000'
MQPSM_COMPAT	1	X'00000001'
MQPSM_ENABLED	2	X'00000002'

MQPSPROP_ * (Właściwości komunikatu publikowania/subskrypcji)

Tabela 279. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSPROP_NONE	0	X'00000000'
MQPSPROP_COMPAT,	1	X'00000001'
MQPSPROP_RFH2	2	X'00000002'
MQPSPROP_MSGPROP,	3	X'00000003'

MQPSST_ * (typ statusu publikowania/subskrypcji dla formatu komendy)

Tabela 280. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPSST_ALL (wszystkie)	0	X'00000000'
MQPSST_LOCAL.	1	X'00000001'
MQPSST_PARENT	2	X'00000002'
MQPSST_CHILD	3	X'00000003'

MQPUBO_ * (opcje publikowania/subskrypcji)

Tabela 281. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPUBO_BRAK	0	X'00000000'
MQPUBO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQPUBO_RETAIN_PUBLIKACJA	2	X'00000002'
MQPUBO_OTHER_SUBSCRIBERS_ONLY	4	X'00000004'
MQPUBO_NO_REGISTRATION	8	X'00000008'
MQPUBO_IS_RETAINED_PUBLIKOWANIE	16	X'00000010'

MQPXP_ * (struktura parametru wyjścia routingu publikowania/subskrypcji)

Tabela 282. Struktury stałych

Nazwa	Struktura
MQPXP_STRUC_ID	"PXP¬"
MQPXP_STRUC_ID_ARRAY	'P','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 283. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQPXP_VERSION_1	1	X'00000001'
MQPXP_CURRENT_VERSION	1	X'00000001'

MQQA_ * (atrybuty kolejki)

Nie zezwalaj na pobieranie wartości

Tabela 284. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQA_GET_INHIBITED	1	X'00000001'
MQQA_GET_ALLOWED	0	X'00000000'

Nie zezwalaj na wstawiane wartości

Tabela 285. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQA_PUT_INHIBITED	1	X'00000001'

Tabela 285. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQA_PUT_ALLOWED	0	X'00000000'

Możliwość współużytkowania kolejki

Tabela 286. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQA_SHAREABLE	1	X'00000001'
Tabela MQQA_NOT_SHAREABLE	0	X'00000000'

Hartowanie z powrotem

Tabela 287. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQA_BACKOUT_HARTOWANE	1	X'00000001'
MQQA_BACKOUT_NOT_HARTOWANE	0	X'00000000'

MQQDT_ * (typy definicji kolejki)

Tabela 288. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQDT_PREDEFINED (predefiniowana)	1	X'00000001'
MQQDT_PERMANENT_DYNAMIC,	2	X'00000002'
MQQDT_TEMPORARY_DYNAMIC	3	X'00000003'
MQQDT_SHARED_DYNAMIC,	4	X'00000004'

MQQF_ * (flagi kolejki)

Tabela 289. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQF_LOCAL_Q,	1	X'00000001'
MQQF_CLWL_USEQ_ANY	64	X'00000040'
MQQF_CLWL_USEQ_LOCAL.	128	X'00000080'

MQQMDT_ * (typy definicji menedżera kolejek w formacie komend)

Tabela 290. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQMDT_JAWNY_NADAWCA_KLASTRA_KLASTRA	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_ * (flagi menedżera kolejek)

Tabela 291. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQMF_REPOSITORY_Q_MGR (menedżer kolejek)	2	X'00000002'
Zdefiniowana przez użytkownika MQQMF_CLUSSDR_USER_DEFINED	8	X'00000008'
MQQMF_CLUSSDR_AUTO_DEFINED	16	X'00000010'
MQQMF_DOSTĘPNE	32	X'00000020'

MQQMFAC_ * (Narzędzie menedżera kolejek w formacie komendy)

Tabela 292. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQMFAC_IMS_BRIDGE,	1	X'00000001'
MQQMFAC_DB2	2	X'00000002'

MQQMSTA_ * (Status menedżera kolejek w formacie komendy)

Tabela 293. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQMSTA_STARTING	1	X'00000001'
MQQMSTA_RUNNING	2	X'00000002'
MQQMSTA_QUIESCING	3	X'00000003'

MQQMT_ * (typy menedżerów kolejek w formacie komend)

Tabela 294. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQMT_NORMAL	0	X'00000000'
MQQMT_REPOSITORY	1	X'00000001'

MQQO_ * (Opcje wyciszania formatu komendy)

Tabela 295. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQO_TAK	1	X'00000001'
MQQO_NIE	0	X'00000000'

MQQSGD_ * (dyspozycje grupy współużytkowania kolejek)

Tabela 296. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQSGD_ALL	-1	X'FFFFFFFF'
MQQSGD_Q_MGR	0	X'00000000'
KOPIOWANA MQQSGD_COPY	1	X'00000001'
MQQSGD_SHARED	2	X'00000002'
GRUPA MQQSGD_GROUP	3	X'00000003'

Tabela 296. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQSGD_PRIVATE,	4	X'00000004'
MQQSGD_LIVE	6	X'00000006'

MQQSGS_ * (Status grupy współużytkowania kolejki w formacie komendy)

Tabela 297. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQSGS_UNKNOWN	0	X'00000000'
MQQSGS_CREATED	1	X'00000001'
MQQSGS_AKTYWNE	2	X'00000002'
MQQSGS_INACTIVE (NIEAKTYWNE)	3	X'00000003'
MQQSGS_FAILED,	4	X'00000004'
MQQSGS_PENDING,	5	X'00000005'

MQQSIE_ * (usługa kolejki w formacie komendy-zdarzenia odstępu czasu)

Tabela 298. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQSIE_BRAK	0	X'00000000'
MQQSIE_HIGH	1	X'00000001'
MQQSIE_OK	2	X'00000002'

MQQSO_ * (Opcje otwierania statusu kolejki dla komend SET, BROWSE, INPUT)

Tabela 299. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQSO_NO	0	X'00000000'
MQQSO_TAK	1	X'00000001'
MQQSO_SHARED	1	X'00000001'
MQQSO_EXCLUSIVE (na wyłączność)	2	X'00000002'

MQQSOT_ * (Format komendy Status kolejki Otwarte typy)

Tabela 300. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQQSOT_ALL	1	X'00000001'
MQQSOT_INPUT	2	X'00000002'
MQQSOT_OUTPUT	3	X'00000003'

MQQSUM_ * (Niezatwierdzone komunikaty statusu kolejki dla formatu komendy)

Tabela 301. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQSUM_TAK	1	X'00000001'
MQQSUM_NO	0	X'00000000'

MQQT_ * (typy kolejek i typy kolejek rozszerzonych)

Typy kolejek

Tabela 302. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQT_LOCAL,	1	X'00000001'
MODEL MQQT	2	X'00000002'
ALIAS MQQT	3	X'00000003'
MQQT_REMOTE	6	X'00000006'
MQQT_CLUSTER	7	X'00000007'

Rozszerzone typy kolejek

Tabela 303. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQQT_ALL	1001	X'000003E9'

MQRC_ * (kody przyczyny)

Tabela 304. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_BRAK	0	X'00000000'
MQRC_APPL_FIRST	900	X'00000384'
MQRC_APPL_LAST	999	X'000003E7'
BŁĄD MQRC_ALIAS_BASE_Q_TYPE_ERROR	2001	X'000007D1'
POŁĄCZONO MQRC_ALREADY_CONNECTED	2002	X'000007D2'
MQRC_BACKED_OUT	2003	X'000007D3'
BŁĄD MQRC_BUFFER_ERROR	2004	X'000007D4'
MQRC_BUFFER_LENGTH_ERROR	2005	X'000007D5'
BŁĄD MQRC_CHAR_ATTR_LENGTH_ERROR	2006	X'000007D6'
BŁĄD MQRC_CHAR_ATTRS_ERROR	2007	X'000007D7'
MQRC_CHAR_ATTRS_TOO_SHORT	2008	X'000007D8'
ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN	2009	X'000007D9'
BŁĄD MQRC_DATA_LENGTH_ERROR	2010	X'000007DA'
BŁĄD MQRC_DYNAMIC_Q_NAME_ERROR	2011	X'000007DB'
BŁĄD MQRC_ENVIRONMENT_ERROR	2012	X'000007DC'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD UTRATY ważności MQRC_EXPIRY_ERROR	2013	X'000007DD'
BŁĄD MQRC_FEEDBACK_ERROR	2014	X'000007DE'
MQRC_GET_INHIBITED	2016	X'000007E0'
MQRC_HANDLE_NOT_AVAILABLE	2017	X'000007E1'
BŁĄD TABELI MQRC_HCONN_ERROR	2018	X'000007E2'
BŁĄD MQRC_HOBJ_ERROR	2019	X'000007E3'
BŁĄD MQRC_INHIBIT_VALUE_ERROR	2020	X'000007E4'
BŁĄD MQRC_INT_ATTR_COUNT_ERROR	2021	X'000007E5'
MQRC_INT_ATTR_COUNT_TOO_SMALL	2022	X'000007E6'
BŁĄD TABELI MQRC_INT_ATTRS_ARRAY_ERROR	2023	X'000007E7'
MQRC_SYNCPOINT_LIMIT_REACHED	2024	X'000007E8'
MQRC_MAX_CONNS_LIMIT_REACHED	2025	X'000007E9'
MQRC_MD_BŁĄD	2026	X'000007EA'
MQRC_MISSING_REPLY_TO_Q	2027	X'000007EB'
BŁĄD 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 (nieautoryzowany)	2035	X'000007F3'
MQRC_NOT_OPEN_FOR_BROWSE,	2036	X'000007F4'
MQRC_NOT_OPEN_FOR_INPUT (MQRC_NOT_OPEN_PUT)	2037	X'000007F5'
MQRC_NOT_OPEN_FOR_INQUIRE (NIEDOSTĘPNE)	2038	X'000007F6'
MQRC_NOT_OPEN_FOR_OUTPUT	2039	X'000007F7'
MQRC_NOT_OPEN_FOR_SET (nieustawione)	2040	X'000007F8'
MQRC_OBJECT_CHANGED	2041	X'000007F9'
MQRC_OBJECT_IN_UŻYTK	2042	X'000007FA'
BŁĄD MQRC_OBJECT_TYPE_ERROR	2043	X'000007FB'
BŁĄD OD_MQRC	2044	X'000007FC'
MQRC_OPTION_NOT_VALID_FOR_TYPE	2045	X'000007FD'
BŁĄD MQRC_OPTIONS_ERROR	2046	X'000007FE'
BŁĄD MQRC_PERSISTENCE_ERROR	2047	X'000007FF'
MQRC_PERSISTENT_NOT_ALLOWED	2048	X'00000800'
MQRC_PRIORITY_PRZEKROCZONE_MAKSIMUM	2049	X'00000801'
BŁĄD MQRC_PRIORITY_ERROR	2050	X'00000802'
MQRC_PUT_INHIBITED	2051	X'00000803'
MQRC_Q_DELETED (usunięto MQRC_Q_)	2052	X'00000804'
MQRC_Q_FULL	2053	X'00000805'
MQRC_Q_NOT_EMPTY	2055	X'00000807'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_Q_SPACE_NOT_AVAILABLE	2056	X'00000808'
BŁĄD MQRC_Q_TYPE_ERROR	2057	X'00000809'
BŁĄD MQRC_Q_MGR_NAME_ERROR	2058	X'0000080A'
MQRC_Q_MGR_NOT_AVAILABLE	2059	X'0000080B'
BŁĄD MQRC_REPORT_OPTIONS_ERROR	2061	X'0000080D'
MQRC_SECOND_MARK_NOT_ALLOWED	2062	X'0000080E'
BŁĄD MQRC_SECURITY_ERROR	2063	X'0000080F'
BŁĄD MQRC_SELECTOR_COUNT_ERROR	2065	X'00000811'
MQRC_SELECTOR_LIMIT_EXCEEDED	2066	X'00000812'
BŁĄD WYWOŁANIA MQRC_SELECTOR_ERROR	2067	X'00000813'
MQRC_SELECTOR_NOT_FOR_TYPE (typ niewymuszenia)	2068	X'00000814'
MQRC_SIGNAL_OCZEKUJĄCE	2069	X'00000815'
MQRC_SIGNAL_REQUEST_ACCEPTED	2070	X'00000816'
MQRC_STORAGE_NIEDOSTĘPNY	2071	X'00000817'
MQRC_SYNCPOINT_NIEDOSTĘPNE	2072	X'00000818'
BŁĄD 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 (błąd typu wyzwalacza)	2078	X'0000081E'
MQRC_TRUNCATED_MSG_ACCEPTED	2079	X'0000081F'
Niepowodzenie 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'
BŁĄD INTERVAL_MQRC_WAIT_ERROR	2090	X'0000082A'
BŁĄD MQRC_XMIT_Q_TYPE_ERROR	2091	X'0000082B'
MQRC_XMIT_Q_USAGE_ERROR,	2092	X'0000082C'
MQRC_NOT_OPEN_FOR_PASS_ALL-WSZYSTKIE	2093	X'0000082D'
MQRC_NOT_OPEN_FOR_PASS_IDENT (identyfikator PASS)	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'
MQRC_OBJECT_USZKODZONA	2101	X'00000835'
MQRC_RESOURCE_PROBLEM	2102	X'00000836'
MQRC_ANOTHER_Q_MGR_CONNECTED	2103	X'00000837'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_UNKNOWN_REPORT_OPCJA	2104	X'00000838'
BŁĄD KLASY MQRC_STORAGE_CLASS_ERROR	2105	X'00000839'
MQRC_COD_NOT_VALID_FOR_XCF_Q	2106	X'0000083A'
MQRC_XWAIT_ANULOWANA	2107	X'0000083B'
BŁĄD MQRC_XWAIT_ERROR	2108	X'0000083C'
MQRC_SUPPRESSED_BY_EXIT	2109	X'0000083D'
BŁĄD MQRC_FORMAT_ERROR	2110	X'0000083E'
BŁĄD MQRC_SOURCE_CCSID_ERROR	2111	X'0000083F'
BŁĄD MQRC_SOURCE_INTEGER_ENC_ERROR	2112	X'00000840'
BŁĄD MQRC_SOURCE_DECIMAL_ENC_ERROR	2113	X'00000841'
MQRC_SOURCE_FLOAT_ENC_ERROR.	2114	X'00000842'
BŁĄD MQRC_TARGET_CCSID_ERROR	2115	X'00000843'
BŁĄD MQRC_TARGET_INTEGER_ENC_ERROR	2116	X'00000844'
BŁĄD MQRC_TARGET_DECIMAL_ENC_ERROR	2117	X'00000845'
MQRC_TARGET_FLOAT_ENC_ERROR	2118	X'00000846'
MQRC_NOT_CONVERTED (nie skonwertowany)	2119	X'00000847'
MQRC_CONVERTED_MSG_TOO_BIG	2120	X'00000848'
MQRC_OBCIĘTE	2120	X'00000848'
MQRC_NO_EXTERNAL_UCZESTNICY	2121	X'00000849'
MQRC_XX_ENCODE_CASE_ONE partia_cząstkowa NIEDOSTĘPNA	2122	X'0000084A'
MQRC_OUTCOME_MIXED	2123	X'0000084B'
MQRC_OUTCOME_PENDING (oczekiwanie na wynik MQ)	2124	X'0000084C'
MQRC_BRIDGE_STARTED,	2125	X'0000084D'
MQRC_BRIDGE_STOPPED	2126	X'0000084E'
MQRC_ADAPTER_STORAGE_NIEDOBÓR	2127	X'0000084F'
MQRC_UOW_IN_PROGRESS	2128	X'00000850'
BŁĄD MQRC_ADAPTER_CONN_LOAD_ERROR	2129	X'00000851'
BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR	2130	X'00000852'
BŁĄD MQRC_ADAPTER_DEFS_ERROR	2131	X'00000853'
MQRC_ADAPTER_DEFS_LOAD_ERROR	2132	X'00000854'
MQRC_ADAPTER_CONV_LOAD_ERROR	2133	X'00000855'
MQRC_BO_BŁĄD	2134	X'00000856'
BŁĄD MQRC_DH_ERROR	2135	X'00000857'
MQRC_MULTIPLE_POWODY	2136	X'00000858'
MQRC_OPEN_FAILED,	2137	X'00000859'
BŁĄD MQRC_ADAPTER_DISC_LOAD_ERROR	2138	X'0000085A'
BŁĄD MQRC_CNO_ERROR	2139	X'0000085B'
MQRC_CICS_WAIT_FAILED	2140	X'0000085C'
BŁĄD MQRC_DLH_ERROR	2141	X'0000085D'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD STERTY MQRC_HEADER_ERROR	2142	X'0000085E'
BŁĄD MQRC_SOURCE_LENGTH_ERROR	2143	X'0000085F'
BŁĄD MQRC_TARGET_LENGTH_ERROR	2144	X'00000860'
BŁĄD MQRC_SOURCE_BUFFER_ERROR	2145	X'00000861'
BŁĄD MQRC_TARGET_BUFFER_ERROR	2146	X'00000862'
BŁĄD MQRC_IIH_ERROR	2148	X'00000864'
BŁĄD MQRC_PCF_ERROR	2149	X'00000865'
BŁĄD MQRC_DBCS_ERROR	2150	X'00000866'
BŁĄD MQRC_OBJECT_NAME_ERROR	2152	X'00000868'
Błąd MQRC_OBJECT_Q_MGR_NAME_ERROR	2153	X'00000869'
MQRC_RECS_PRESENT_ERROR	2154	X'0000086A'
BŁĄD REKORDÓW MQRC_OBJECT_RECORDS	2155	X'0000086B'
BŁĄD MQRC_RESPONSE_RECORDS_ERROR	2156	X'0000086C'
MQRC_ASID_MISMATCH	2157	X'0000086D'
MQRC_PMO_RECORD_FLAGS_ERROR	2158	X'0000086E'
BŁĄD REKORDÓW MQRC_PUT_MSG_RECORDS	2159	X'0000086F'
ID_KOND_MQRC_W_UŻYCIU	2160	X'00000870'
MQRC_Q_MGR_QUIESCING,	2161	X'00000871'
MQRC_Q_MGR_ZATRZYMYWANIE	2162	X'00000872'
MQRC_DUPLICATE_RECOV_COORD	2163	X'00000873'
BŁĄD MQRC_PMO_ERROR	2173	X'0000087D'
MQRC_API_EXIT_NOT_FOUND (nie znaleziono)	2182	X'00000886'
BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR	2183	X'00000887'
BŁĄD NAZWY MQRC_REMOTE_Q_NAME_ERROR	2184	X'00000888'
MQRC_INCONSISTENT_PERSISTENCE	2185	X'00000889'
BŁĄD MQRC_GMO_ERROR	2186	X'0000088A'
MQRC_CICS_BRIDGE_RESTRICTION;	2187	X'0000088B'
MQRC_STOPPED_BY_CLUSTER_EXIT,	2188	X'0000088C'
BŁĄD MQRC_CLUSTER_RESOLUTION_ERROR	2189	X'0000088D'
MQRC_CONVERTED_STRING_TOO_BIG	2190	X'0000088E'
BŁĄD MQRC_TMC_ERROR	2191	X'0000088F'
MQRC_PAGESET_FULL	2192	X'00000890'
MQRC_STORAGE_MEDIUM_FULL	2192	X'00000890'
BŁĄD STRONY MQRC_PAGESET_ERROR	2193	X'00000891'
MQRC_NAME_NOT_VALID_FOR_TYPE (nazwa nie jest poprawna)	2194	X'00000892'
BŁĄD MQRC_UNEXPECTED_ERROR	2195	X'00000893'
MQRC_UNKNOWN_XMIT_Q	2196	X'00000894'
MQRC_UNKNOWN_DEF_XMIT_Q	2197	X'00000895'
BŁĄD MQRC_DEF_XMIT_Q_TYPE_ERROR	2198	X'00000896'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_DEF_XMIT_Q_USAGE_ERROR,	2199	X'00000897'
MQRC_MSG_MARKED_BROWSE_CO_OP	2200	X'00000898'
MQRC_NAME_IN_USE (nazwa użytkownika MQRC)	2201	X'00000899'
MQRC_CONNECTION_QUIESCING,	2202	X'0000089A'
ZATRZYMANE_POŁĄCZENIA_MQRC	2203	X'0000089B'
MQRC_ADAPTER_NIEDOSTĘPNE	2204	X'0000089C'
BŁĄD MQRC_MSG_ID_ERROR	2206	X'0000089E'
BŁĄD MQRC_CORREL_ID_ERROR	2207	X'0000089F'
BŁĄD SYSTEMU MQRC_FILE_	2208	X'000008A0'
MQRC_NO_MSG_LOCKED,	2209	X'000008A1'
BŁĄD MQRC_SOAP_DOTNET_	2210	X'000008A2'
BŁĄD MQRC_SOAP_AXIS_ERROR	2211	X'000008A3'
BŁĄD MQRC_SOAP_URL_ERROR	2212	X'000008A4'
PLIK MQRC_NOT_AUDITED	2216	X'000008A8'
MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)	2217	X'000008A9'
MQRC_MSG_TOO_BIG_FOR_CHANNEL	2218	X'000008AA'
MQRC_CALL_W_TOKU	2219	X'000008AB'
BŁĄD MQRC_RMH_ERROR	2220	X'000008AC'
MQRC_Q_MGR_AKTYWNE	2222	X'000008AE'
MQRC_Q_MGR_NOT_ACTIVE (nieaktywna)	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'
BŁĄD MQRC_RFH_HEADER_FIELD_ERROR	2228	X'000008B4'
BŁĄD WŁAŚCIWOŚCI MQRC_RAS_PROPERTY_ERROR	2229	X'000008B5'
MQRC_UNIT_OF_WORK_NOT_STARTED (nie uruchomiono)	2232	X'000008B8'
MQRC_CHANNEL_AUTO_DEF_OK	2233	X'000008B9'
MQRC_CHANNEL_AUTO_DEF_BŁĄD	2234	X'000008BA'
BŁĄD MQRC_CFH_ERROR	2235	X'000008BB'
BŁĄD MQRC_CFIL_ERROR	2236	X'000008BC'
BŁĄD MQRC_CFIN_	2237	X'000008BD'
BŁĄD MQRC_CFSL_ERROR	2238	X'000008BE'
MQRC_CFST_BŁĄD	2239	X'000008BF'
MQRC_INCOMPLETE_GROUP	2241	X'000008C1'
MQRC_INCOMPLETE_MSG	2242	X'000008C2'
Identyfikatory MQRC_INCONSISTENT_CCSID	2243	X'000008C3'
MQRC_INCONSISTENT_ENCODINGS,	2244	X'000008C4'
MQRC_INCONSISTENT_UOW,	2245	X'000008C5'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_INVALID_MSG_UNDER_CURSOR,	2246	X'000008C6'
BŁĄD MQRC_MATCH_OPTIONS_ERROR	2247	X'000008C7'
BŁĄD MQRC_MDE_ERROR	2248	X'000008C8'
BŁĄD MQRC_MSG_FLAGS_ERROR	2249	X'000008C9'
BŁĄD MQRC_MSG_SEQ_NUMBER_ERROR	2250	X'000008CA'
BŁĄD MQRC_OFFSET_	2251	X'000008CB'
BŁĄD MQRC_ORIGINAL_LENGTH_ERROR	2252	X'000008CC'
MQRC_SEGMENT_LENGTH_ZERO	2253	X'000008CD'
MQRC_UOW_NIEDOSTĘPNE	2255	X'000008CF'
MQRC_WRONG_GMO_VERSION	2256	X'000008D0'
MQRC_WRONG_MD_VERSION	2257	X'000008D1'
BŁĄD MQRC_GROUP_ID_ERROR	2258	X'000008D2'
MQRC_INCONSISTENT_BROWSE (przeglądanie niespójne)	2259	X'000008D3'
BŁĄD MQRC_XQH_ERROR	2260	X'000008D4'
BŁĄD WYWOŁANIA MQRC_SRC_ENV_ERROR	2261	X'000008D5'
BŁĄD MQRC_SRC_NAME_ERROR	2262	X'000008D6'
MQRC_DEST_ENV_ERROR	2263	X'000008D7'
BŁĄD MQRC_DEST_NAME_ERROR	2264	X'000008D8'
BŁĄD MQRC_TM_ERROR	2265	X'000008D9'
BŁĄD MQRC_CLUSTER_EXIT_ERROR	2266	X'000008DA'
BŁĄD ŁADOWANIA MQRC_CLUSTER_EXIT_LOAD_ERROR	2267	X'000008DB'
MQRC_CLUSTER_PUT_INHIBITED	2268	X'000008DC'
BŁĄD ZASOBU MQRC_CLUSTER_RESOURCE_ERROR	2269	X'000008DD'
MQRC_NO_DESTINATIONS_AVAILABLE	2270	X'000008DE'
MQRC_CONN_TAG_IN_UŻYJ	2271	X'000008DF'
MQRC_PARTIALLY_CONVERTED	2272	X'000008E0'
BŁĄD POŁĄCZENIA MQRC_CONNECTION_	2273	X'000008E1'
BŁĄD MQRC_OPTION_ENVIRONMENT_ERROR	2274	X'000008E2'
BŁĄD MQRC_CD_ERROR	2277	X'000008E5'
MQRC_CLIENT_CONN_BŁĄD	2278	X'000008E6'
MQRC_CHANNEL_STOPPED_BY_USER	2279	X'000008E7'
BŁĄD MQRC_HCONFIG_ERROR	2280	X'000008E8'
BŁĄD FUNKCJI MQRC_FUNCTION_	2281	X'000008E9'
MQRC_CHANNEL_STARTED	2282	X'000008EA'
MQRC_CHANNEL_STOPPED	2283	X'000008EB'
MQRC_CHANNEL_CONV_ERROR	2284	X'000008EC'
MQRC_SERVICE_NOT_AVAILABLE	2285	X'000008ED'
Niepowodzenie MQRC_INITIALIZATION_FAILED	2286	X'000008EE'
Niepowodzenie MQRC_TERMINATION_FAILED	2287	X'000008EF'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_UNKNOWN_Q_NAME	2288	X'000008F0'
BŁĄD MQRC_SERVICE_ERROR	2289	X'000008F1'
MQRC_Q_ALREADY_EXISTS	2290	X'000008F2'
ID_UŻYTKOWNIKA_MQRC_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_ANULOWANA	2297	X'000008F9'
MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)	2298	X'000008FA'
BŁĄD TYPU MQRC_SELECTOR_TYPE_ERROR	2299	X'000008FB'
BŁĄD MQRC_COMMAND_TYPE_ERROR	2300	X'000008FC'
MQRC_MULTIPLE_INSTANCE_ERROR BŁĄD	2301	X'000008FD'
MQRC_SYSTEM_ITEM_NOT_ALTERABLE (nie można)	2302	X'000008FE'
BŁĄD KONWERSJI MQRC_BAG_CONVERSION_ERROR	2303	X'000008FF'
MQRC_SELECTOR_OUT_OF_RANGE	2304	X'00000900'
MQRC_SELECTOR_NOT_UNIQUE (NIEUNIKALNY)	2305	X'00000901'
MQRC_INDEX_NOT_PRESENT (NIEOBECNY)	2306	X'00000902'
BŁĄD KOMENDY MQRC_STRING_ERROR	2307	X'00000903'
MQRC_ENCODING_NOT_SUPPORTED	2308	X'00000904'
MQRC_SELECTOR_NOT_PRESENT (NIEOBECNY)	2309	X'00000905'
BŁĄD WYBORU MQRC_OUT_SELECTOR_ERROR	2310	X'00000906'
MQRC_STRING_OBCIĘTE	2311	X'00000907'
MQRC_SELECTOR_WRONG_TYPE	2312	X'00000908'
MQRC_INCONSISTENT_ITEM_TYPE,	2313	X'00000909'
BŁĄD MQRC_INDEX_ERROR	2314	X'0000090A'
MQRC_SYSTEM_BAG_NOT_ALTERABLE (nieprzełączalny)	2315	X'0000090B'
BŁĄD MQRC_ITEM_COUNT_ERROR	2316	X'0000090C'
MQRC_FORMAT_NOT_SUPPORTED	2317	X'0000090D'
MQRC_SELECTOR_NOT_SUPPORTED	2318	X'0000090E'
BŁĄD MQRC_ITEM_VALUE_ERROR	2319	X'0000090F'
BŁĄD MQRC_HBAG_ERROR	2320	X'00000910'
BRAK PARAMETRU MQRC_PARAMETER_MISSING	2321	X'00000911'
MQRC_CMD_SERVER_NOT_AVAILABLE (niedostępna)	2322	X'00000912'
BŁĄD MQRC_STRING_LENGTH_ERROR	2323	X'00000913'
MQRC_INQUIRY_COMMAND_ERROR, błąd	2324	X'00000914'
MQRC_NESTED_BAG_NOT_SUPPORTED (nieobsługiwana)	2325	X'00000915'
MQRC_BAG_WRONG_TYPE	2326	X'00000916'

Tabela 304. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
BŁĄD MQRC_ITEM_TYPE_ERROR	2327	X'00000917'
MQRC_SYSTEM_BAG_NOT_DELETABLE (nie można usunąć)	2328	X'00000918'
MQRC_SYSTEM_ITEM_NOT_DELETABLE (nie można usunąć)	2329	X'00000919'
BŁĄD MQRC_CODED_CHAR_SET_ID_ERROR	2330	X'0000091A'
MQRC_MSG_TOKEN_ERROR,	2331	X'0000091B'
MQRC_MISSING_WIH (brak łącznika MQRC)	2332	X'0000091C'
BŁĄD MQRC_WIH_ERROR	2333	X'0000091D'
BŁĄD MQRC_RFH_ERROR	2334	X'0000091E'
BŁĄD WYWOŁANIA MQRC_RFH_STRING_ERROR	2335	X'0000091F'
BŁĄD MQRC_RFH_COMMAND_ERROR	2336	X'00000920'
BŁĄD MQRC_RFH_PARM_ERROR	2337	X'00000921'
MQRC_RFH_DUPLICATE_PARM	2338	X'00000922'
BRAK MQRC_RFH_PARM_MISSING	2339	X'00000923'
BŁĄD KONWERSJI 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 (obiekt MQRC_NOT_UNIQUE)	2343	X'00000927'
MQRC_CONN_TAG_NOT_RELEASED	2344	X'00000928'
MQRC_CF_NIEDOSTĘPNE	2345	X'00000929'
MQRC_CF_STRUC_IN_UŻYJ	2346	X'0000092A'
MQRC_CF_STRUC_LIST_HDR_IN_USE	2347	X'0000092B'
MQRC_CF_STRUC_AUTH_FAILED	2348	X'0000092C'
MQRC_CF_STRUC_BŁĄD	2349	X'0000092D'
MQRC_CONN_TAG_NOT_USABLE (nieużywany)	2350	X'0000092E'
MQRC_GLOBAL_UOW_CONFLICT	2351	X'0000092F'
MQRC_LOCAL_UOW_CONFLICT (konflikt uow_rejestru)	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'
BŁĄD MQRC_WXP_ERROR	2356	X'00000934'
BŁĄD MQRC_CURRENT_RECORD_ERROR	2357	X'00000935'
BŁĄD MQRC_NEXT_OFFSET_	2358	X'00000936'
MQRC_NO_RECORD_AVAILABLE	2359	X'00000937'
MQRC_OBJECT_LEVEL_NIEKOMPATYBILNY	2360	X'00000938'
BŁĄD MQRC_NEXT_RECORD_ERROR	2361	X'00000939'
MQRC_BACKOUT_THRESHOLD_REACHED	2362	X'0000093A'
MQRC_MSG_NOT_MATCHED (komunikat MQRC)	2363	X'0000093B'
BŁĄD MQRC_JMS_FORMAT_ERROR	2364	X'0000093C'
MQRC_SEGMENTS_NOT_SUPPORTED (nieobsługiwane)	2365	X'0000093D'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_WRONG_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_PARTANT_NOT_DEFINED (nie zdefiniowano IPANT_MQRC)	2372	X'00000944'
MQRC_CF_STRUC_FAILED	2373	X'00000945'
BŁĄD MQRC_API_EXIT_ERROR	2374	X'00000946'
MQRC_API_EXIT_INIT_BŁĄD	2375	X'00000947'
BŁĄD MQRC_API_EXIT_TERM_ERROR	2376	X'00000948'
BŁĄD MQRC_EXIT_REASON_ERROR	2377	X'00000949'
BŁĄD MQRC_RESERVED_VALUE_ERROR	2378	X'0000094A'
MQRC_NO_DATA_AVAILABLE	2379	X'0000094B'
BŁĄD ZASIĘGU MQRC_SCO_ERROR	2380	X'0000094C'
BŁĄD MQRC_KEY_REPOSITORY_ERROR	2381	X'0000094D'
BŁĄD MQRC_CRYPTO_HARDWARE_ERROR	2382	X'0000094E'
BŁĄD MQRC_AUTH_INFO_REC_COUNT_ERROR	2383	X'0000094F'
MQRC_AUTH_INFO_REC_BŁĄD	2384	X'00000950'
BŁĄD MQRC_AIR_ERROR	2385	X'00000951'
BŁĄD MQRC_AUTH_INFO_TYPE_ERROR	2386	X'00000952'
BŁĄD MQRC_AUTH_INFO_CONN_NAME_ERROR	2387	X'00000953'
BŁĄD MQRC_LDAP_USER_NAME_ERROR	2388	X'00000954'
MQRC_LDAP_USER_NAME_LENGTH_ERR	2389	X'00000955'
BŁĄD 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 (błąd)	2393	X'00000959'
BŁĄD MQRC_Q_INDEX_TYPE_ERROR	2394	X'0000095A'
BŁĄD MQRC_CFBS_ERROR	2395	X'0000095B'
MQRC_SSL_NOT_ALLOWED (NIEWŁĄCZONY)	2396	X'0000095C'
BŁĄD MQRC_JSSE_ERROR	2397	X'0000095D'
MQRC_SSL_PEER_NAME_MISMATCH	2398	X'0000095E'
MQRC_SSL_PEER_NAME_ERROR BŁĄD	2399	X'0000095F'
MQRC_UNSUPPORTED_CIPHER_SUITE,	2400	X'00000960'
MQRC_SSL_CERTIFICATE_ODWOŁANO	2401	X'00000961'
BŁĄD MQRC_SSL_CERT_STORE_ERROR	2402	X'00000962'
BŁĄD ŁADOWANIA MQRC_CLIENT_EXIT_LOAD_ERROR	2406	X'00000966'
BŁĄD WYJŚCIA MQRC_CLIENT_EXIT_ERROR	2407	X'00000967'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRC_UOW_ZATWIERDZ.	2408	X'00000968'
MQRC_SSL_KEY_RESET_BŁĄD	2409	X'00000969'
MQRC_UNKNOWN_COMPONENT_NAME	2410	X'0000096A'
STATUS MQRC_LOGGER_STATUS	2411	X'0000096B'
MQRC_COMMAND_MQSC	2412	X'0000096C'
MQRC_COMMAND_PCF,	2413	X'0000096D'
MQRC_CFIF_BŁĄD	2414	X'0000096E'
BŁĄD MQRC_CFSF_ERROR	2415	X'0000096F'
BŁĄD MQRC_CFGR_ERROR	2416	X'00000970'
MQRC_MSG_NOT_ALLOWED_IN_GROUP	2417	X'00000971'
BŁĄD MQRC_FILTER_OPERATOR_ERROR	2418	X'00000972'
BŁĄD MQRC_NESTED_SELECTOR_ERROR	2419	X'00000973'
BŁĄD MQRC_EPH_ERROR	2420	X'00000974'
BŁĄD MQRC_RFH_FORMAT_ERROR	2421	X'00000975'
MQRC_CFBF_BŁĄD	2422	X'00000976'
MQRC_CLIENT_CHANNEL_CONFLICT	2423	X'00000977'
BŁĄD MQRC_SD_ERROR	2424	X'00000978'
BŁĄD MQRC_TOPIC_STRING_ERROR	2425	X'00000979'
BŁĄD MQRC_STS	2426	X'0000097A'
MQRC_NO_SUBSCRIPTION	2428	X'0000097C'
MQRC_SUBSCRIPTION_IN_USE,	2429	X'0000097D'
BŁĄD TYPU MQRC_STAT_TYPE_ERROR	2430	X'0000097E'
MQRC_SUB_USER_DATA_BŁĄD	2431	X'0000097F'
MQRC_SUB_ALREADY_EXISTS	2432	X'00000980'
MQRC_IDENTITY_MISMATCH	2434	X'00000982'
BŁĄD PODRZĘDNY MQRC_ALTER_SUB_ERROR	2435	X'00000983'
MQRC_DURABILITY_NOT_ALLOWED	2436	X'00000984'
MQRC_NO_RETAINED_MSG	2437	X'00000985'
BŁĄD MQRC_SRO_ERROR	2438	X'00000986'
BŁĄD MQRC_SUB_NAME_ERROR	2440	X'00000988'
BŁĄD WYWOŁANIA MQRC_OBJECT_STRING_ERROR	2441	X'00000989'
BŁĄD MQRC_PROPERTY_NAME_ERROR	2442	X'0000098A'
MQRC_SEGMENTATION_NOT_ALLOWED	2443	X'0000098B'
BŁĄD MQRC_CBD_ERROR	2444	X'0000098C'
BŁĄD MQRC_CTLO_ERROR	2445	X'0000098D'
Aktywne wywołania zwrotne MQRC_NO_CALLBACKS_ACTIVE	2446	X'0000098E'
MQRC_CALLBACK_NOT_REGISTERED	2448	X'00000990'
MQRC_OPTIONS_CHANGED	2457	X'00000999'
MQRC_READ_AHEAD_MSGS	2458	X'0000099A'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD SYNTAX_MQRC_SELECTOR_SYNTAX_ERROR	2459	X'0000099B'
BŁĄD MQRC_HMSG_ERROR	2460	X'0000099C'
BŁĄD MQRC_CMHO_ERROR	2461	X'0000099D'
BŁĄD MQRC_DMHO_ERROR	2462	X'0000099E'
BŁĄD MQRC_SMPO_ERROR	2463	X'0000099F'
BŁĄD 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'
WŁAŚCIWOŚĆ_MQRC_NIEDOSTĘPNA	2471	X'000009A7'
MQRC_PROP_NUMBER_FORMAT_BŁĄD	2472	X'000009A8'
BŁĄD TYPU WŁAŚCIWOŚCI MQRC_PROPERTY_TYPE_ERROR	2473	X'000009A9'
MQRC_PROPERTIES_TOO_BIG	2478	X'000009AE'
MQRC_PUT_NOT_ZACHOWANE	2479	X'000009AF'
MQRC_ALIAS_TARGTYPE_CHANGED	2480	X'000009B0'
BŁĄD MQRC_DMPO_ERROR	2481	X'000009B1'
BŁĄD MQRC_PD_ERROR	2482	X'000009B2'
BŁĄD MQRC_CALLBACK_TYPE_ERROR	2483	X'000009B3'
BŁĄD MQRC_CBD_OPTIONS_ERROR	2484	X'000009B4'
MQRC_MAX_MSG_LENGTH_ERROR	2485	X'000009B5'
BŁĄD MQRC_CALLBACK_ROUTINE_ERROR	2486	X'000009B6'
BŁĄD MQRC_CALLBACK_LINK_ERROR	2487	X'000009B7'
BŁĄD OPERACJI MQRC	2488	X'000009B8'
BŁĄD MQRC_BMHO_ERROR	2489	X'000009B9'
Właściwość MQRC_UNSUPPORTED_PROPERTY	2490	X'000009BA'
MQRC_PROP_NAME_NOT_CONVERTED (nieskonwertowany)	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 (nie dozwolona)	2498	X'000009C2'
MQRC_MSG_HANDLE_IN_USE,	2499	X'000009C3'
MQRC_HCONN_ASYNC_AKTYWNE	2500	X'000009C4'
BŁĄD MQRC_MHBO_ERROR	2501	X'000009C5'
MQRC_PUBLICATION_FAILURE	2502	X'000009C6'
MQRC_SUB_INHIBITED	2503	X'000009C7'
MQRC_SELECTOR_ALWAYS_FALSE	2504	X'000009C8'

Tabela 304. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD MQRC_XEPO_ERROR	2507	X'000009CB'
MQRC_DURABILITY_NOT_ALTERABLE	2509	X'000009CD'
MQRC_TOPIC_NOT_ALTERABLE (nie MOŻNA)	2510	X'000009CE'
MQRC_SUBLEVEL_NOT_ALTERABLE (nie MOŻNA)	2512	X'000009D0'
MQRC_PROPERTY_NAME_LENGTH_ERR	2513	X'000009D1'
MQRC_DUPLICATE_GROUP_SUB	2514	X'000009D2'
MQRC_GROUPING_NOT_ALTERABLE (grupa MQRC_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'
Błąd łańcucha MQRC_SELECTION_STRING_ERROR	2519	X'000009D7'
MQRC_RES_OBJECT_STRING_BŁĄD	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'
BŁĄD MQRC_RETAINED_MSG_Q_ERROR	2525	X'000009DD'
MQRC_RETAINED_NOT_DOSTARCZONE	2526	X'000009DE'
MQRC_RFH_RESTRICTED_FORMAT_ERR	2527	X'000009DF'
MQRC_CONNECTION_ZATRZYMANY	2528	X'000009E0'
MQRC_ASYNC_UOW_CONFLICT	2529	X'000009E1'
MQRC_ASYNC_XA_CONFLICT	2530	X'000009E2'
MQRC_PUBSUB_INHIBITED	2531	X'000009E3'
MQRC_MSG_HANDLE_COPY_FAILURE	2532	X'000009E4'
MQRC_DEST_CLASS_NOT_ALTERABLE (klasa_REST)	2533	X'000009E5'
MQRC_OPERATION_NOT_ALLOWED (operacja MQRC_NOT_ALLOWED)	2534	X'000009E6'
BŁĄD DZIAŁANIA MQRC_ACTION_	2535	X'000009E7'
MQRC_CHANNEL_NOT_AVAILABLE	2537	X'000009E9'
MQRC_HOST_NOT_AVAILABLE (niedostępny)	2538	X'000009EA'
BŁĄD MQRC_CHANNEL_CONFIG_ERROR	2539	X'000009EB'
MQRC_UNKNOWN_CHANNEL_NAME	2540	X'000009EC'
MQRC_LOOPING_PUBLIKOWANIE	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'
BŁĄD ELEMENTU MQRC_SUITE_B_ERROR	2592	X'00000A20'
BŁĄD MQRC_PASSWORD_PROTECTION_ERROR	2594	X'00000A22'

Tabela 304. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
BŁĄD MQRC_REOPEN_EXCL_INPUT_ERROR	6100	X'000017D4'
BŁĄD MQRC_REOPEN_INQUIRE_ERROR	6101	X'000017D5'
MQRC_REOPEN_SAVED_CONTEXT_ERR	6102	X'000017D6'
BŁĄD MQRC_REOPEN_TEMPORARY_Q_ERROR	6103	X'000017D7'
MQRC_ATTRIBUTE_LOCKED (atrybut mqrc_locked)	6104	X'000017D8'
MQRC_CURSOR_NOT_VALID (MQRC_CURSOR_NOT_VALID	6105	X'000017D9'
BŁĄD MQRC_ENCODING_ERROR	6106	X'000017DA'
BŁĄD MQRC_STRUC_ID_ERROR	6107	X'000017DB'
MQRC_NULL_POINTER (wskaźnik wartości NULL)	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_INSUFFICIENT_BUFFER (MQRC_INC_BUFFER)	6113	X'000017E1'
MQRC_INSUFFICIENT_DATA	6114	X'000017E2'
MQRC_DATA_OBCIĘTE	6115	X'000017E3'
MQRC_ZERO_LENGTH	6116	X'000017E4'
MQRC_NEGATIVE_LENGTH DŁUGOŚĆ	6117	X'000017E5'
MQRC_NEGATIVE_OFFSET	6118	X'000017E6'
MQRC_INCONSISTENT_FORMAT,	6119	X'000017E7'
MQRC_INCONSISTENT_OBJECT_STATE,	6120	X'000017E8'
MQRC_CONTEXT_OBJECT_NOT_VALID	6121	X'000017E9'
BŁĄD MQRC_CONTEXT_OPEN_ERROR	6122	X'000017EA'
MQRC_STRUC_LENGTH_ERROR	6123	X'000017EB'
MQRC_NIE_POŁĄCZONO	6124	X'000017EC'
MQRC_NOT_OPEN (nieotwarte)	6125	X'000017ED'
MQRC_DISTRIBUTION_LIST_EMPTY	6126	X'000017EE'
MQRC_INCONSISTENT_OPEN_OPTIONS,	6127	X'000017EF'
MQRC_WRONG_VERSION	6128	X'000017F0'
BŁĄD MQRC_REFERENCE_ERROR	6129	X'000017F1'

MQRCCF_ * (Kody przyczyny nagłówka formatu komendy)

Więcej informacji na temat odpowiedzi programisty zawiera sekcja Kody przyczyny PCF.

Tabela 305. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
BŁĄD MQRCCF_CFH_TYPE_ERROR	3001	X'00000BB9'
BŁĄD MQRCCF_CFH_LENGTH_ERROR	3002	X'00000BBA'
MQRCCF_CFH_VERSION_ERROR	3003	X'00000BBB'
MQRCCF_CFH_MSG_SEQ_NUMBER_ERR	3004	X'00000BBC'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD MQRCCF_CFH_CONTROL_ERROR	3005	X'00000BBD'
MQRCCF_CFH_PARM_COUNT_BŁĄD	3006	X'00000BBE'
BŁĄD MQRCCF_CFH_COMMAND_ERROR	3007	X'00000BBF'
MQRCCF_COMMAND_FAILED	3008	X'00000BC0'
MQRCCF_CFIN_LENGTH_ERROR	3009	X'00000BC1'
BŁĄD MQRCCF_CFST_LENGTH_ERROR	3010	X'00000BC2'
MQRCCF_CFST_STRING_LENGTH_ERR	3011	X'00000BC3'
BŁĄD MQRCCF_FORCE_VALUE_ERROR	3012	X'00000BC4'
BŁĄD MQRCCF_STRUCTURE_TYPE_ERROR	3013	X'00000BC5'
BŁĄD MQRCCF_CFIN_PARM_ID_ERROR	3014	X'00000BC6'
BŁĄD 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'
BŁĄD MQRCCF_Q_TYPE_ERROR	3022	X'00000BCE'
BŁĄD MQRCCF_MD_FORMAT_ERROR	3023	X'00000BCF'
BŁĄD MQRCCFSL_CFSL_LENGTH_ERROR	3024	X'00000BD0'
BŁĄD MQRCCF_REPLACE_VALUE_ERROR	3025	X'00000BD1'
MQRCCF_CFIL_DUPLICATE_VALUE	3026	X'00000BD2'
BŁĄD MQRCCF_CFIL_COUNT_ERROR	3027	X'00000BD3'
MQRCCF_CFIL_LENGTH_ERROR	3028	X'00000BD4'
BŁĄD MQRCCF_QUIESCE_VALUE_ERROR	3029	X'00000BD5'
BŁĄD MQRCCF_MODE_VALUE_ERROR	3029	X'00000BD5'
BŁĄD MQRCCF_MSG_SEQ_NUMBER_ERROR	3030	X'00000BD6'
BŁĄD MQRCCF_PING_DATA_COUNT_ERROR	3031	X'00000BD7'
MQRCCF_PING_DATA_COMPARE_ERROR	3032	X'00000BD8'
BŁĄD MQRCCF_CFSL_PARM_ID_ERROR	3033	X'00000BD9'
BŁĄD MQRCCF_CHANNEL_TYPE_ERROR	3034	X'00000BDA'
MQRCCF_PARM_SEQUENCE_ERROR BŁĄD	3035	X'00000BDB'
MQRCCF_XMIT_PROTOCOL_TYPE_ERR	3036	X'00000BDC'
BŁĄD MQRCCF_BATCH_SIZE_ERROR	3037	X'00000BDD'
MQRCCF_DISC_INT_BŁĄD	3038	X'00000BDE'
MQRCCF_SHORT_RETRY_ERROR (BŁĄD WYWOŁANIA MQRCCF)	3039	X'00000BDF'
BŁĄD MQRCCF_SHORT_TIMER_ERROR	3040	X'00000BE0'
MQRCCF_LONG_RETRY_ERROR (błąd odtwarzania)	3041	X'00000BE1'
BŁĄD MQRCCF_LONG_TIMER_ERROR	3042	X'00000BE2'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_SEQ_NUMBER_WRAP_BŁĄD	3043	X'00000BE3'
BŁĄD MQRCCF_MAX_MSG_LENGTH_ERROR	3044	X'00000BE4'
MQRCCF_PUT_AUTH_BŁĄD	3045	X'00000BE5'
BŁĄD MQRCCF_PURGE_VALUE_ERROR	3046	X'00000BE6'
BŁĄD MQRCCF_CFIL_PARM_ID_ERROR	3047	X'00000BE7'
MQRCCF_MSG_OBCIĘTE	3048	X'00000BE8'
BŁĄD MQRCCF_CCSID_ERROR	3049	X'00000BE9'
MQRCCF_ENCODING_BŁĄD	3050	X'00000BEA'
MQRCCF_QUEUES_VALUE_ERROR	3051	X'00000BEB'
BŁĄD MQRCCF_DATA_CONV_VALUE_ERROR	3052	X'00000BEC'
BŁĄD MQRCCF_INDOUBT_VALUE_ERROR	3053	X'00000BED'
BŁĄD MQRCCF_ESCAPE_TYPE_ERROR	3054	X'00000BEE'
BŁĄD MQRCCF_REPOS_VALUE_ERROR	3055	X'00000BEF'
BŁĄD MQRCCF_CHANNEL_TABLE_ERROR	3062	X'00000BF6'
BŁĄD MQRCCF_MCA_TYPE_ERROR	3063	X'00000BF7'
BŁĄD 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'
BŁĄD MQRCCF_CFSL_COUNT_ERROR	3068	X'00000BFC'
MQRCCF_CFSL_STRING_LENGTH_ERR	3069	X'00000BFD'
MQRCCF_BROKER_USUNIĘTE	3070	X'00000BFE'
BŁĄD STRUMIENIA MQRCCF_STREAM_ERROR	3071	X'00000BFF'
BŁĄD MQRCCF_TOPIC_ERROR	3072	X'00000C00'
MQRCCF_NOT_REGISTERED (niezarejestrowane)	3073	X'00000C01'
BŁĄD MQRCCF_Q_MGR_NAME_ERROR	3074	X'00000C02'
MQRCCF_INCORRECT_STREAM	3075	X'00000C03'
BŁĄD 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'
BŁĄD MQRCCF_CORREL_ID_ERROR	3080	X'00000C08'
MQRCCF_NOT_AUTHORIZED (nieautoryzowany)	3081	X'00000C09'
MQRCCF_UNKNOWN_STREAM	3082	X'00000C0A'
BŁĄD MQRCCF_REG_OPTIONS_ERROR	3083	X'00000C0B'
MQRCCF_PUB_OPTIONS_ERROR	3084	X'00000C0C'
MQRCCF_UNKNOWN_BROKER	3085	X'00000C0D'
BŁĄD MQRCCF_Q_MGR_CCSID_ERROR	3086	X'00000C0E'
BŁĄD MQRCCF_DEL_OPTIONS_ERROR	3087	X'00000C0F'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_CLUSTER_NAME_CONFLICT (konflikt nazwa_klastra_mqrc)	3088	X'00000C10'
Konflikt MQRCCF_REPOS_NAME_CONFLICT	3089	X'00000C11'
MQRCCF_CLUSTER_Q_USAGE_ERROR,	3090	X'00000C12'
BŁĄD MQRCCF_ACTION_VALUE_ERROR	3091	X'00000C13'
BŁĄD BIBLIOTEKI MQRCCF_COMMS_LIBRARY_ERROR	3092	X'00000C14'
BŁĄD MQRCCF_NETBIOS_NAME_ERROR	3093	X'00000C15'
Komenda MQRCCF_BROKER_COMMAND_FAILED	3094	X'00000C16'
MQRCCF_CFST_CONFLICTING_PARM (obiekt MQRCCF_CONFLICTING_PARM)	3095	X'00000C17'
MQRCCF_PATH_NOT_VALID (NIEPOPRAWNE)	3096	X'00000C18'
MQRCCF_PARM_SYNTAX_ERROR	3097	X'00000C19'
MQRCCF_PWD_LENGTH_ERROR	3098	X'00000C1A'
MQRCCF_FILTER_BŁĄD	3150	X'00000C4E'
MQRCCF_WRONG_USER	3151	X'00000C4F'
MQRCCF_DUPLICATE_SUBSCRIPTION	3152	X'00000C50'
BŁĄD MQRCCF_SUB_NAME_ERROR	3153	X'00000C51'
BŁĄD MQRCCF_SUB_IDENTITY_ERROR	3154	X'00000C52'
MQRCCF_SUBSCRIPTION_IN_USE	3155	X'00000C53'
MQRCCF_SUBSCRIPTION_LOCKED (zablokowana subskrypcja MQ)	3156	X'00000C54'
MQRCCF_ALREADY_JOINED	3157	X'00000C55'
MQRCCF_XX_ENCODE_CASE_ONE obiekt_w_użyciu	3160	X'00000C58'
MQRCCF_UNKNOWN_FILE_NAME	3161	X'00000C59'
MQRCCF_FILE_NOT_AVAILABLE	3162	X'00000C5A'
BŁĄD MQRCCF_DISC_RETRY_ERROR	3163	X'00000C5B'
BŁĄD MQRCCF_ALLOC_RETRY_ERROR	3164	X'00000C5C'
BŁĄD MQRCCF_ALLOC_SLOW_TIMER_ERROR	3165	X'00000C5D'
BŁĄD MQRCCF_ALLOC_FAST_TIMER_ERROR	3166	X'00000C5E'
BŁĄD MQRCCF_PORT_NUMBER_ERROR	3167	X'00000C5F'
MQRCCF_CHL_SYSTEM_NOT_ACTIVE (nie aktywny)	3168	X'00000C60'
BRAK MQRCCF_ENTITY_NAME_MISSING	3169	X'00000C61'
MQRCCF_PROFILE_NAME_ERROR	3170	X'00000C62'
BŁĄD MQRCCF_AUTH_VALUE_ERROR	3171	X'00000C63'
MQRCCF_AUTH_WARTOŚĆ_BRAK	3172	X'00000C64'
BRAK MQRCCF_OBJECT_TYPE_MISSING	3173	X'00000C65'
BŁĄD MQRCCF_CONNECTION_ID_ERROR	3174	X'00000C66'
BŁĄD MQRCCF_LOG_TYPE_ERROR	3175	X'00000C67'
MQRCCF_PROGRAM_NIEDOSTĘPNE	3176	X'00000C68'
MQRCCF_PROGRAM_AUTH_FAILED-niepowodzenie	3177	X'00000C69'
MQRCCF_NONE_FOUND (znaleziono nieodebrane pakiety MQ)	3200	X'00000C80'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_SECURITY_SWITCH_OFF,	3201	X'00000C81'
Niepowodzenie MQRCCF_SECURITY_REFRESH_FAILED	3202	X'00000C82'
MQRCCF_PARM_CONFLICT (konflikt parametrów MQ)	3203	X'00000C83'
MQRCCF_COMMAND_INHIBITED	3204	X'00000C84'
MQRCCF_OBJECT_BEING_DELETED	3205	X'00000C85'
MQRCCF_STORAGE_CLASS_IN_UŻYJ	3207	X'00000C87'
MQRCCF_OBJECT_NAME_RESTRICTED	3208	X'00000C88'
MQRCCF_OBJECT_LIMIT_EXCEEDED	3209	X'00000C89'
MQRCCF_OBJECT_OPEN_FORCE	3210	X'00000C8A'
MQRCCF_DISPOSITION_CONFLICT (konflikt dyspozycji MQ)	3211	X'00000C8B'
MQRCCF_Q_MGR_NOT_IN_QSG	3212	X'00000C8C'
MQRCCF_ATTR_VALUE_FIXED	3213	X'00000C8D'
BŁĄD LISTY MQRCCF_NAME_ERROR	3215	X'00000C8F'
MQRCCF_NO_CHANNEL_INITIATOR	3217	X'00000C91'
BŁĄD MQRCCF_CHANNEL_INITIATOR_ERROR	3218	X'00000C92'
MQRCCF_COMMAND_LEVEL_CONFLICT (konflikt poziomu komend)	3222	X'00000C96'
MQRCCF_Q_ATTR_CONFLICT (konflikt atrybutów kolejki MQ)	3223	X'00000C97'
MQRCCF_EVENTS_DISABLED	3224	X'00000C98'
BŁĄD MQRCCF_COMMAND_SCOPE_ERROR	3225	X'00000C99'
BŁĄD MQRCCF_COMMAND_REPLY_ERROR	3226	X'00000C9A'
MQRCCF_FUNCTION_RESTRICTED,	3227	X'00000C9B'
BRAK PARAMETRU MQRCCF_PARM_MISSING	3228	X'00000C9C'
BŁĄD MQRCCF_PARM_VALUE_ERROR	3229	X'00000C9D'
MQRCCF_COMMAND_LENGTH_ERROR	3230	X'00000C9E'
BŁĄD MQRCCF_COMMAND_ORIGIN_ERROR	3231	X'00000C9F'
MQRCCF_LISTENER_CONFLICT (konflikt odbiornika nasłuchiwania MQ)	3232	X'00000CA0'
MQRCCF_LISTENER_STARTED (uruchomione)	3233	X'00000CA1'
MQRCCF_LISTENER_ZATRZYMANY	3234	X'00000CA2'
BŁĄD MQRCCF_CHANNEL_ERROR	3235	X'00000CA3'
MQRCCF_CF_STRUC_ERROR (błąd MQRCCF_STRUC_ERROR)	3236	X'00000CA4'
MQRCCF_UNKNOWN_USER_ID	3237	X'00000CA5'
MQRCCF_UNEXPECTED_ERROR (nieoczekiwany błąd MQ)	3238	X'00000CA6'
MQRCCF_NO_XCF_PARTNER	3239	X'00000CA7'
MQRCCF_CFGR_PARM_ID_BŁĄD	3240	X'00000CA8'
BŁĄD MQRCCF_CFIF_LENGTH_ERROR	3241	X'00000CA9'
BŁĄD MQRCCF_CFIF_OPERATOR_ERROR	3242	X'00000CAA'
BŁĄD MQRCCF_CFIF_PARM_ID_ERROR	3243	X'00000CAB'
MQRCCF_CFSF_FILTER_VAL_LEN_ERR	3244	X'00000CAC'
BŁĄD MQRCCF_CFSF_LENGTH_ERROR	3245	X'00000CAD'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD MQRCCF_CFSF_OPERATOR_ERROR	3246	X'00000CAE'
BŁĄD 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 (zatrzymano usługę MQ)	3253	X'00000CB5'
MQRCCF_CFBS_DUPLICATE_PARM	3254	X'00000CB6'
BŁĄD MQRCCF_CFBS_LENGTH_ERROR	3255	X'00000CB7'
BŁĄD MQRCCF_CFBS_PARM_ID_ERROR	3256	X'00000CB8'
MQRCCF_CFBS_STRING_LENGTH_ERR	3257	X'00000CB9'
BŁĄD MQRCCF_CFGR_LENGTH_ERROR	3258	X'00000CBA'
MQRCCF_CFGR_PARM_COUNT_BŁĄD	3259	X'00000CBB'
MQRCCF_CONN_NOT_STOPPED, komenda	3260	X'00000CBC'
MQRCCF_SERVICE_REQUEST_PENDING (oczekiwanie na żądanie usługi MQ)	3261	X'00000CBD'
MQRCCF_NO_START_CMD	3262	X'00000CBE'
MQRCCF_NO_STOP_CMD	3263	X'00000CBF'
BŁĄD MQRCCF_CFBF_LENGTH_ERROR	3264	X'00000CC0'
BŁĄD MQRCCF_CFBF_PARM_ID_ERROR	3265	X'00000CC1'
MQRCCF_CFBF_BŁĄD OPERATOR_ERROR	3266	X'00000CC2'
MQRCCF_CFBF_FILTER_VAL_LEN_ERR	3267	X'00000CC3'
MQRCCF_LISTENER_STILL_AKTYWNE	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_CONFLICT (mqrccf_konflikt przypadków zabezpieczeń)	3303	X'00000CE7'
BŁĄD MQRCCF_TOPIC_TYPE_ERROR	3305	X'00000CE9'
BŁĄD 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_BŁĄD_PUNKTU_SUBSKRYPCJI	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'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BŁĄD MQRCCF_DEST_NAME_ERROR	3316	X'00000CF4'
MQRCCF_INVALID_DESTINATION	3317	X'00000CF5'
MQRCCF_PUBSUB_INHIBITED	3318	X'00000CF6'
BŁĄD TYPU MQRCCF_CHLAUTH_TYPE_ERROR	3326	X'00000CFE'
BŁĄD DZIAŁANIA MQRCCF_CHLAUTH_ACTION_	3327	X'00000CFF'
MQRCCF_CHLAUTH_USERSRC_BŁĄD	3335	X'00000D07'
MQRCCF_WRONG_CHLAUTH_TYPE	3336	X'00000D08'
MQRCCF_CHLAUTH_ALREADY_EXISTS	3337	X'00000D09'
MQRCCF_CHLAUTH_NOT_FOUND	3338	X'00000D0A'
MQRCCF_WRONG_CHLAUTH_DZIAŁANIE	3339	X'00000D0B'
MQRCCF_WRONG_CHLAUTH_USERSRC	3340	X'00000D0C'
MQRCCF_CHLAUTH_WARN_BŁĄD	3341	X'00000D0D'
MQRCCF_WRONG_CHLAUTH_MATCH	3342	X'00000D0E'
MQRCCF_IPADDR_RANGE_CONFLICT	3343	X'00000D0F'
MQRCCF_CHLAUTH_MAX_EXCEEDED	3344	X'00000D10'
MQRCCF_IPADDR_BŁĄD	3345	X'00000D11'
BŁĄD MQRCCF_IPADDR_RANGE_ERROR	3346	X'00000D12'
BRAK MQRCCF_PROFILE_NAME_MISSING	3347	X'00000D13'
MQRCCF_CHLAUTH_CLNTUSER_ERROR (błąd MQRCCF_CHLAUTH_CLNTUSER_	3348	X'00000D14'
BŁĄD MQRCCF_CHLAUTH_NAME_ERROR	3349	X'00000D15'
MQRCCF_SUITE_ERROR	3353	X'00000D19'
MQRCCF_PSCLUS_DISABLED_TOPDEF,	3359	X'00000D1F'
MQRCCF_PSCLUS_TOPIC_EXISTS	3360	X'00000D20'
MQRCCF_INVALID_PROTOCOL	3365	X'00000D25'
MQRCCF_ACCESS_BLOCKED	3382	X'00000D36'
MQRCCF_OBJECT_ALREADY_EXISTS	4001	X'00000FA1'
MQRCCF_OBJECT_WRONG_TYPE	4002	X'00000FA2'
MQRCCF_LIKE_OBJECT_WRONG_TYPE (mqrccf_typ_błędnego_obiektu)	4003	X'00000FA3'
MQRCCF_OBJECT_OPEN	4004	X'00000FA4'
BŁĄD MQRCCF_ATTR_VALUE_ERROR	4005	X'00000FA5'
MQRCCF_UNKNOWN_Q_MGR	4006	X'00000FA6'
MQRCCF_Q_WRONG_TYPE	4007	X'00000FA7'
BŁĄD MQRCCF_OBJECT_NAME_ERROR	4008	X'00000FA8'
MQRCCF_ALLOCATE_FAILED	4009	X'00000FA9'
MQRCCF_HOST_NOT_AVAILABLE	4010	X'00000FAA'
MQRCCF_XX_ENCODE_CASE_ONE błąd_konfiguracji	4011	X'00000FAB'
MQRCCF_CONNECTION_ODMOWA	4012	X'00000FAC'
MQRCCF_ENTRY_ERROR	4013	X'00000FAD'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_SEND_FAILED	4014	X'00000FAE'
BŁĄD MQRCCF_RECEIVED_DATA_ERROR	4015	X'00000FAF'
MQRCCF_RECEIVE_XX_ENCODE_CASE_ONE niepowodzenie	4016	X'00000FB0'
MQRCCF_CONNECTION_CLOSED	4017	X'00000FB1'
MQRCCF_NO_STORAGE	4018	X'00000FB2'
MQRCCF_NO_COMMS_MANAGER	4019	X'00000FB3'
MQRCCF_LISTENER_NOT_STARTED (nie uruchomiono)	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_PING_BŁĄD	4030	X'00000FBE'
MQRCCF_CHANNEL_IN_USE	4031	X'00000FBF'
MQRCCF_CHANNEL_NOT_FOUND	4032	X'00000FC0'
MQRCCF_UNKNOWN_REMOTE_CHANNEL (nieznany kanał zdalny)	4033	X'00000FC1'
MQRCCF_REMOTE_QM_NIEDOSTĘPNE	4034	X'00000FC2'
MQRCCF_REMOTE_QM_TERMINATING	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_NIEDOSTĘPNE	4039	X'00000FC7'
MQRCCF_COMMIT_FAILED	4040	X'00000FC8'
MQRCCF_WRONG_CHANNEL_TYPE	4041	X'00000FC9'
MQRCCF_CHANNEL_ALREADY_EXISTS	4042	X'00000FCA'
MQRCCF_DATA_TOO_LARGE	4043	X'00000FCB'
BŁĄD MQRCCF_CHANNEL_NAME_ERROR	4044	X'00000FCC'
BŁĄD MQRCCF_XMIT_Q_NAME_ERROR	4045	X'00000FCD'
BŁĄD MQRCCF_MCA_NAME_ERROR	4047	X'00000FCF'
BŁĄD MQRCCF_SEND_EXIT_NAME_ERROR	4048	X'00000FD0'
BŁĄD MQRCCF_SEC_EXIT_NAME_ERROR	4049	X'00000FD1'
BŁĄD MQRCCF_MSG_EXIT_NAME_ERROR	4050	X'00000FD2'
BŁĄD MQRCCF_RCV_EXIT_NAME_ERROR	4051	X'00000FD3'
MQRCCF_XMIT_Q_NAME_WRONG_TYPE	4052	X'00000FD4'
MQRCCF_NAZWA_MCA_WRONG_TYPE	4053	X'00000FD5'
MQRCCF_DISC_INT_WRONG_TYPE	4054	X'00000FD6'
MQRCCF_SHORT_RETRY_WRONG_TYPE	4055	X'00000FD7'
MQRCCF_SHORT_TIMER_WRONG_TYPE	4056	X'00000FD8'

Tabela 305. Wartości stałych (kontynuacja)

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCCF_LONG_RETRY_WRONG_TYPE	4057	X'00000FD9'
MQRCCF_LONG_TIMER_WRONG_TYPE	4058	X'00000FDA'
MQRCCF_PUT_AUTH_WRONG_TYPE	4059	X'00000FDB'
MQRCCF_KEEP_ALIVE_INT_BŁĄD	4060	X'00000FDC'
MQRCCF_MISSING_CONN_NAME (brakująca_nazwa_konn)	4061	X'00000FDD'
BŁĄD MQRCCF_CONN_NAME_ERROR	4062	X'00000FDE'
MQRCCF_MQSET_FAILED	4063	X'00000FDF'
MQRCCF_CHANNEL_NOT_ACTIVE (nieaktywne)	4064	X'00000FE0'
MQRCCF_TERMINATED_BY_SEC_EXIT	4065	X'00000FE1'
BŁĄD MQRCCF_DYNAMIC_Q_SCOPE_ERROR	4067	X'00000FE3'
MQRCCF_CELL_DIR_NOT_AVAILABLE	4068	X'00000FE4'
BŁĄD MQRCCF_MR_COUNT_ERROR	4069	X'00000FE5'
MQRCCF_MR_COUNT_WRONG_TYPE	4070	X'00000FE6'
BŁĄD MQRCCF_MR_EXIT_NAME_ERROR	4071	X'00000FE7'
MQRCCF_MR_EXIT_NAME_WRONG_TYPE (typ błędu wyjścia MQ)	4072	X'00000FE8'
MQRCCF_MR_INTERVAL_ERROR	4073	X'00000FE9'
MQRCCF_MR_INTERVAL_WRONG_TYPE	4074	X'00000FEA'
MQRCCF_NPM_SPEED_BŁĄD	4075	X'00000FEB'
MQRCCF_NPM_SPEED_WRONG_TYPE	4076	X'00000FEC'
MQRCCF_HB_INTERVAL_ERROR (BŁĄD INTERFEJSU MQRCCF)	4077	X'00000FED'
MQRCCF_HB_INTERVAL_WRONG_TYPE	4078	X'00000FEE'
MQRCCF_CHAD_BŁĄD	4079	X'00000FEF'
MQRCCF_CHAD_WRONG_TYPE	4080	X'00000FF0'
BŁĄD MQRCCF_CHAD_EVENT_ERROR	4081	X'00000FF1'
MQRCCF_CHAD_EVENT_WRONG_TYPE	4082	X'00000FF2'
BŁĄD MQRCCF_CHAD_EXIT_ERROR	4083	X'00000FF3'
MQRCCF_CHAD_EXIT_WRONG_TYPE	4084	X'00000FF4'
MQRCCF_SUPPRESSED_BY_EXIT	4085	X'00000FF5'
BŁĄD MQRCCF_BATCH_INT_	4086	X'00000FF6'
MQRCCF_BATCH_INT_WRONG_TYPE	4087	X'00000FF7'
MQRCCF_NET_PRIORITY_ERROR	4088	X'00000FF8'
MQRCCF_NET_PRIORITY_WRONG_TYPE	4089	X'00000FF9'
MQRCCF_CHANNEL_CLOSED	4090	X'00000FFA'
MQRCCF_Q_STATUS_NOT_FOUND	4091	X'00000FFB'
BŁĄD MQRCCF_SSL_CIPHER_SPEC_ERROR	4092	X'00000FFC'
BŁĄD MQRCCF_SSL_PEER_NAME_ERROR	4093	X'00000FFD'
MQRCCF_SSL_CLIENT_AUTH_ERROR	4094	X'00000FFE'
MQRCCF_RETAINED_NOT_SUPPORTED (brak obsługi)	4095	X'00000FFF'
z/OS MQRCCF_KWD_VALUE_WRONG_TYPE	4096	X'00001000'

MQRCN_ * (Stałe ponownego połączenia klienta)

Tabela 306. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCN_NO	0	X'00000000'
MQRCN_YES	1	X'00000001'
MQRCN_Q_MGR	2	X'00000002'
MQRCN_DISABLED	3	X'00000003'

MQRCVTIME_ * (typy limitów czasu odbierania)

Tabela 307. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRCVTIME_MULTIPLY	0	X'00000000'
MQRCVTIME_ADD	1	X'00000001'
MQRCVTIME_EQUAL	2	X'00000002'

MQREADA_ * (wartości odczytu z wyprzedzeniem)

Tabela 308. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQREADA_NO	0	X'00000000'
MQREADA_YES	1	X'00000001'
MQREADA_WYŁĄCZONA	2	X'00000002'
MQREADA_INHIBITED	3	X'00000003'
MQREADA_BACKLOG,	4	X'00000004'

MQRECORDING_ * (opcje rejestrowania)

Tabela 309. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRECORDING_DISABLED,	0	X'00000000'
MQRECORDING_Q (kolejka MQ)	1	X'00000001'
MQRECORDING_MSG	2	X'00000002'

MQREGO_ * (opcje rejestracji publikowania/subskrypcji)

Tabela 310. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQREGO_BRAK	0	X'00000000'
MQREGO_CORREL_ID_AS_IDENTITY	1	X'00000001'
MQREGO_ANONYMOUS	2	X'00000002'
MQREGO_XX_ENCODE_CASE_ONE lokalne	4	X'00000004'
MQREGO_DIRECT_REQUESTS (żądania przekierowania)	8	X'00000008'
MQREGO_NOWA_PUBLIKACJA	16	X'00000010'
MQREGO_PUBLISH_ON_REQUEST_ONLY,	32	X'00000020'

Tabela 310. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQREGO_DEREGISTER_ALL	64	X'00000040'
MQREGO_INCLUDE_STREAM_NAME	128	X'00000080'
MQREGO_INFORM_IF_ZACHOWANY	256	X'00000100'
MQREGO_DUPLICATES_OK	512	X'00000200'
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_ALTERATION	32768	X'00008000'
MQREGO_FULL_RESPONSE	65536	X'00010000'
MQREGO_JOIN_SHARED	131072	X'00020000'
MQREGO_JOIN_EXCLUSIVE	262144	X'00040000'
MQREGO_LEAVE_ONLY (TYLKO_MQREG)	524288	X'00080000'
ID_UŻYTKOWNIKA MQREGO_VARIABLE_USER_ID	1048576	X'00100000'
MQREGO_LOCKED,	2097152	X'00200000'

MQRFH_ * (struktura i flagi nagłówka reguł i formatowania)

Reguły i struktura nagłówka formatowania

Tabela 311. Struktury stałych	
Nazwa	**Struktura**
MQRFH_ID_struktury	"RFH¬"
MQRFH_STRUC_ID_ARRAY	'R','F','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 312. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'

Flagi nagłówka reguł i formatowania

Tabela 313. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQRFH_BRAK	0	X'00000000'
MQRFH_NO_FLAGS	0	X'00000000'

MQRFH2_* (znacznik opcji publikowania/subskrypcji RFH2 -znaczniki folderu najwyższego poziomu)

Tabela 314. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRFH2_NAME_VALUE_VERSION	1	X'00000001'

MQRFH2_* (nazwy znaczników opcji publikowania/subskrypcji)

MQRFH2_PUBSUB_CMD_FOLDER	"psc"
MQRFH2_PUBSUB_RESP_FOLDER	"pscr"
MQRFH2_MSG_CONTENT_FOLDER	"mcd"
MQRFH2_USER_FOLDER	"usr"

MQRFH2_* (nazwy znaczników XML w opcjach publikowania/subskrypcji)

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_ * (zwrócona długość)

Tabela 315. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRL_UNDEFINED (niezdefiniowana)	-1	X'FFFFFFFF'

MQRMH_ * (struktura nagłówka komunikatu odniesienia)

Tabela 316. Struktury stałych

Nazwa	Struktura
MQRMH_STRUC_ID (ID struktury MQRM)	"RMH¬"
MQRMH_STRUC_ID_ARRAY	'R','M','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 317. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRMH_VERSION_1	1	X'00000001'
MQRMH_CURRENT_VERSION	1	X'00000001'

MQRMHF_ * (flagi nagłówka komunikatu odniesienia)

Tabela 318. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRMHF_LAST	1	X'00000001'
MQRMHF_NOT_LAST (nie)	0	X'00000000'

MQRO_ * (opcje raportu)

Tabela 319. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRO_EXCEPTION,	16777216	X'01000000'
MQRO_EXCEPTION_WITH_DATA	50331648	X'03000000'
MQRO_EXCEPTION_WITH_FULL_DATA	117440512	X'07000000'
MQRO_UTR_WAŻN.	2097152	X'00200000'
MQRO_EXPIRATION_WITH_DATA	6291456	X'00600000'
MQRO_EXPIRATION_WITH_FULL_DATA	14680064	X'00E00000'
MQRO_COA	256	X'00000100'
MQRO_KOA_WITH_DATA	768	X'00000300'
MQRO_KOA_WITH_FULL_DATA	1792	X'00000700'
MQRO_COD	2048	X'00000800'
MQRO_KOD_WITH_DATA	6144	X'00001800'
MQRO_KOD_WITH_FULL_DATA	14336	X'00003800'
MQRO_PAN	1	X'00000001'
MQRO_NAN	2	X'00000002'
DZIAŁANIE 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 (Identyfikator KORELACJI MQ)	64	X'00000040'
MQRO_DEAD_LETTER_Q	0	X'00000000'
MQRO_DISCARD_MSG	134217728	X'08000000'
MQRO_PASS_DISCARD_AND_WAŻNOŚCI	16384	X'00004000'
MQRO_BRAK	0	X'00000000'

*MQRO_ * (maski opcji raportu)*

Tabela 320. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRO_REJECT_UNSUP_MASK (maska_odrzucenia_mqr)	270270464	X'101C0000'
MQRO_ACCEPT_UNSUP_MASK	-270532353	X'EFE000FF'
MQRO_ACCEPT_UNSUP_IF_XMIT_MASK	261888	X'0003FF00'

MQROUTE_ * (Trasa śledzenia)

Maksymalna liczba działań trasy śledzenia (MQIACF_MAX_ACTIVITIES)

Tabela 321. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_UNLIMITED_ACTIVITIES	0	X'00000000'

Szczegóły trasy śledzenia (MQIACF_ROUTE_DETAIL)

Tabela 322. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_DETAIL_LOW	2	X'00000002'
MQROUTE_DETAIL_MEDIUM	8	X'00000008'
MQROUTE_DETAIL_HIGH	32	X'00000020'

Przekazywanie trasy śledzenia (MQIACF_ROUTE_FORWARDING)

Tabela 323. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_FORWARD_ALL	256	X'00000100'
MQROUTE_FORWARD_IF_SUPPORTED.	512	X'00000200'
MQROUTE_FORWARD_REJ_UNSUP_MASK	-65536	X'FFFF0000'

Dostarczanie trasy śledzenia (MQIACF_ROUTE_DELIVERY)

Tabela 324. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_DELIVER_YES	4096	X'00001000'
MQROUTE_DELIVER_NO	8192	X'00002000'
MQROUTE_DELIVER_REJ_UNSUP_MASK,	-65536	X'FFFF0000'

Kumulacja trasy śledzenia (MQIACF_ROUTE_AKUMULACJA)

Tabela 325. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQROUTE_AKUMULATE_NONE	65539	X'00010003'
MQROUTE_AKUMULATE_W_KOMUNIKACIE	65540	X'00010004'
MQROUTE_AKUMULATE_AND_REPLY,	65541	X'00010005'

MQRP_ * (Opcje zastępowania formatu komendy)

Tabela 326. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRP_TAK	1	X'00000001'
MQRP_NO	0	X'00000000'

MQRQ_ * (Kwalifikatory przyczyny formatu komendy)

Tabela 327. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQRQ_CONN_NOT_AUTHORIZED (brak autoryzacji połączenia MQ)	1	X'00000001'
MQRQ_OPEN_NOT_AUTHORIZED (nieautoryzowany)	2	X'00000002'
MQRQ_CLOSE_NOT_AUTHORIZED (nieautoryzowany)	3	X'00000003'
MQRQ_CMD_NOT_AUTHORIZED (nieautoryzowany)	4	X'00000004'
MQRQ_Q_MGR_ZATRZYMYWANIE	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'
BŁĄD MQRQ_BRIDGE_STOPPED_ERROR	12	X'0000000C'
BŁĄD MQRQ_SSL_HANDSHAKE_ERROR	13	X'0000000D'
BŁĄD MQRQ_SSL_CIPHER_SPEC_ERROR	14	X'0000000E'
MQRQ_SSL_CLIENT_AUTH_ERROR	15	X'0000000F'
MQRQ_SSL_PEER_NAME_ERROR BŁĄD	16	X'00000010'
MQRQ_SUB_NOT_AUTHORIZED (nieautoryzowany)	17	X'00000011'
MQRQ_SUB_DEST_NOT_AUTHORIZED	18	X'00000012'
MQRQ_SSL_UNKNOWN_REVOCATION	19	X'00000013'
MQRQ_SYS_CONN_NOT_AUTHORIZED (nieautoryzowany)	20	X'00000014'
MQRQ_CHANNEL_BLOCKED_ADDRESS	21	X'00000015'
MQRQ_CHANNEL_BLOCKED_USERID,	22	X'00000016'
MQRQ_CHANNEL_BLOCKED_NOACCESS	23	X'00000017'
MQRQ_MAX_ACTIVE_CHANNELS	24	X'00000018'
MQRQ_MAX_CHANNELS	25	X'00000019'
MQRQ_SVRCONN_INST_LIMIT	26	X'0000001A'
MQRQ_CLIENT_INST_LIMIT!	27	X'0000001B'
MQRQ_CAF_NOT_INSTALLED	28	X'0000001C'
MQRQ_CSP_NOT_AUTHORIZED (nieautoryzowany)	29	X'0000001D'
MQRQ_FAILOVER_DOZWOLONE	30	X'0000001E'
MQRQ_FAILOVER_NOT_PERMITTED (nie TRWAŁY)	31	X'0000001F'
MQRQ_STANDBY_ACTIVATED MQRQ_STANDBY_	32	X'00000020'
MQRQ_REPLICA_ACTIVATED	33	X'00000021'

MQRT_ * (Typy odświeżania formatu komendy)

Tabela 328. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
KONFIGURACJA_MQR	1	X'00000001'

Tabela 328. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQRT_XX_ENCODE_CASE_ONE wygaśnięcie	2	X'00000002'
MQRT_NSPROC	3	X'00000003'
MQRT_PROXYSUB	4	X'00000004'

MQRU_ * (tylko żądanie)

Tabela 329. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQRU_PUBLISH_ON_REQUEST	1	X'00000001'
MQRU_PUBLISH_ALL	2	X'00000002'

MQSCA_ * (Uwierzytelnianie klienta TLS)

Tabela 330. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSCA_XX_ENCODE_CASE_ONE wymagane	0	X'00000000'
MQSCA_OPCJONALNA	1	X'00000001'

MQSCO_ * (opcje konfiguracyjne TLS)

Struktura opcji konfiguracyjnych TLS

Tabela 331. Struktury stałych	
Nazwa	**Struktura**
MQSCO_STRUC_ID	"SCO¬"
MQSCO_STRUC_ID_ARRAY	'S','C','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 332. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_CURRENT_VERSION	4	X'00000004'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Liczba resetowanych kluczy opcji konfiguracyjnych TLS

Tabela 333. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSCO_RESET_COUNT_DEFAULT	0	X'00000000'

Zasięg definicji kolejki formatu komendy

Tabela 334. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCO_Q_MGR	1	X'00000001'
MQSCO_CELL	2	X'00000002'

MQSCOPE_ * (zasięg publikowania)

Tabela 335. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCOPE_ALL	0	X'00000000'
MQSCOPE_AS_PARENT,	1	X'00000001'
MQSCOPE_QMGR,	4	X'00000004'

MQSCYC_ * (przypadek zabezpieczeń)

Tabela 336. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSCYC_UPPER	0	X'00000000'
MQSCYC_MIXED	1	X'00000001'

MQSD_ * (struktura deskryptora obiektu)

Tabela 337. Stałe nazwy i struktury

Nazwa	Struktura
MQSD_STRUC_ID (identyfikator struktury MQS)	"SD¬¬"
MQSD_STRUC_ID_ARRAY	'S','D','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 338. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSD_VERSION_1	1	X'00000001'
MQSD_CURRENT_VERSION	1	X'00000001'

MQSECITEM_ * (Elementy zabezpieczeń w formacie komendy)

Tabela 339. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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'

Tabela 339. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSECITEM_MXNLIST	8	X'00000008'
MQSECITEM_MXPROC	9	X'00000009'
MQSECITEM_MXQUEUE	10	X'0000000A'
MQSECITEM_MXTOPIC,	11	X'0000000B'

MQSECPROT_ * (typy protokołów zabezpieczeń)

Tabela 340. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSECPROT_BRAK	0	X'00000000'
MQSECPROT_SSLV30	1	X'00000001'
MQSECPROT_TLSV10	2	X'00000002'
MQSECPROT_TLSV12	4	X'00000004'

MQSECSW_ * (Przełączniki i stany przełączników w formacie komendy)

Przełączniki zabezpieczeń formatu komendy

Tabela 341. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
PROCES_MQSECSW_PROCESS	1	X'00000001'
Lista MQSECSW_NAMELIST	2	X'00000002'
MQSECSW_Q	3	X'00000003'
MQSECSW_TOPIC	4	X'00000004'
MQSECSW_CONTEXT	6	X'00000006'
MQSECSW_ALTERNATE_USER,	7	X'00000007'
KOMENDA MQSECSW_COMMAND	8	X'00000008'
POŁĄCZENIE MQSECSW_CONNECTION	9	X'00000009'
PODSYSTEM MQSECSW_SYSTEM	10	X'0000000A'
Zasoby komend MQSECSW_COMMAND_RESOURCES	11	X'0000000B'
MQSECSW_Q_MGR	15	X'0000000F'
MQSECSW_QSG	16	X'00000010'

Stany przełącznika bezpieczeństwa formatu komendy

Tabela 342. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'
BŁĄD MQSECSW_OFF_ERROR	25	X'00000019'
MQSECSW_ON_PRZESŁONIĘTE	26	X'0000001A'

MQSECTYPE_ * (Typy zabezpieczeń formatu komendy)

Tabela 343. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSECTYPE_AUTHSERV	1	X'00000001'
MQSECTYPE_SSL	2	X'00000002'
KLASY MQSECTYPE_CLASSES	3	X'00000003'

MQSEG_ * (segmentacja)

Tabela 344. Nazwy i wartości stałych

Nazwa	Wartość
MQSEG_INHIBITED	'¬'
DOZWOLONY_SEGMENT_MQB	'A'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQSEL_ * (specjalne wartości selektora)

Tabela 345. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSEL_ANY_SELECTOR	-30001	X'FFFF8ACF'
MQSEL_ANY_USER_SELECTOR	-30002	X'FFFF8ACE'
MQSEL_ANY_SYSTEM_SELECTOR (selektor systemu MQ)	-30003	X'FFFF8ACD'
MQSEL_ALL_SELECTORS	-30001	X'FFFF8ACF'
MQSEL_ALL_USER_SELECTORS	-30002	X'FFFF8ACE'
MQSEL_ALL_SYSTEM_SELECTORS,	-30003	X'FFFF8ACD'

MQSELTYPE_ * (typy selektorów)

Tabela 346. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
BRAK MQSELTYPE_NONE	0	X'00000000'
MQSELTYPE_STANDARD	1	X'00000001'
MQSELTYPE_EXTENDED,	2	X'00000002'

MQSID_ * (identyfikator zabezpieczeń)

Tabela 347. Nazwy i wartości stałych

Nazwa	Wartość
MQSID_BRAK	X'00...00' (40 wartości pustych)
MQSID_NONE_ARRAY	'\0','\0',... (40 wartości pustych)

MQSIDT_ * (typy identyfikatorów zabezpieczeń)

Tabela 348. Nazwy i wartości stałych

Nazwa	Wartość szesnastkowa
MQSIDT_BRAK	X'00'

Tabela 348. Nazwy i wartości stałych (kontynuacja)	
Nazwa	**Wartość szesnastkowa**
MQSIDT_NT_SECURITY_ID	X'01'
MQSIDT_WAS_SECURITY_ID	X'02'

MQSMPO_ * (ustawienie opcji i struktury właściwości komunikatu)

Ustaw strukturę opcji właściwości komunikatu

Tabela 349. Struktury stałych	
Nazwa	**Struktura**
MQSMPO_STRUC_ID	"SMPO"
MQSMPO_STRUC_ID_ARRAY	'S','M','P','O'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 350. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSMPO_VERSION_1	1	X'00000001'
MQSMPO_CURRENT_VERSION	1	X'00000001'

Ustaw opcje właściwości komunikatu

Tabela 351. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_BRAK	0	X'00000000'

MQSO_ * (opcje subskrypcji)

Tabela 352. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSO_NONE	0	X'00000000'
MQSO_NON_DURABLE	0	X'00000000'
MQSO_READ_AHEAD_AS_Q_DEF,	0	X'00000000'
MQSO_ALTER (zmiana MQSO)	1	X'00000001'
MQSO_CREATE	2	X'00000002'
MQSO_RESUME	4	X'00000004'
MQSO_DURABLE	8	X'00000008'
MQSO_GROUP_SUB	16	X'00000010'
MQSO_MANAGED	32	X'00000020'
MQSO_SET_IDENTITY_CONTEXT,	64	X'00000040'

Tabela 352. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSO_FIXED_USERID (Identyfikator stałego użytkownika)	256	X'00000100'
MQSO_ANY_USERID	512	X'00000200'
Publikacje MQSO_ON_REQUEST	2048	X'00000800'
MQSO_NEW_PUBLIC TYLKO	4096	X'00001000'
MQSO_FAIL_IF_QUIESCING,	8192	X'00002000'
MQSO_ALTERNATE_UPRAWNIENIA_UŻYTKOWNIKA	262144	X'00040000'
MQSO_WILDCARD_CHAR	1048576	X'00100000'
MQSO_WILDCARD_TOPIC,	2097152	X'00200000'
MQSO_SET_CORREL_ID (Identyfikator korelacji MQSO_SET_)	4194304	X'00400000'
MQSO_SCOPE_QMGR,	67108864	X'04000000'
MQSO_NO_READ_AHEAD	134217728	X'08000000'
MQSO_READ_AHEAD	268435456	X'10000000'

MQSP_ * (dostępność punktu synchronizacji)

Tabela 353. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSP_DOSTĘPNE	1	X'00000001'
MQSP_NIEDOSTĘPNE	0	X'00000000'

MQSPL_ * (Opcje ochrony strategii bezpieczeństwa)

Tabela 354. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSPL_PASSTHRU	0	X'00000000'
MQSPL_REMOVE	1	X'00000001'
MQSPL_AS_POLICY	2	X'00000002'

MQSQQM_ * (nazwa menedżera kolejek współużytkowanych)

Tabela 355. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSQQM_UŻYJ	0	X'00000000'
MQSQQM_IGNORE,	1	X'00000001'

MQSR_ * (działanie)

Tabela 356. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
PUBLIKACJA_DZIAŁANIA_MQSRR	1	X'00000001'

MQSRO_ * (struktura opcji żądań subskrypcji)

Tabela 357. Struktury stałych	
Nazwa	**Struktura**
MQSRO_ID_struktury	"SRO¬"
MQSRO_STRUC_ID_ARRAY	'S','R','O','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 358. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSRO_VERSION_1	1	X'00000001'
MQSRO_CURRENT_VERSION	1	X'00000001'
MQSRO_BRAK	0	X'00000000'
MQSRO_FAIL_IF_QUIESCING,	8192	X'00002000'

MQSS_ * (Status segmentu)

Tabela 359. Stałe nazwy i struktury	
Nazwa	**Struktura**
MQSS_NOT_A_SEGMENT (MQSS_NOT_A)	'¬'
SEGMENT_MQSS	'S'
MQSS_OSTATNI_SEGMENT	'L'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

MQSSL_ * (Wymagania FIPS TLS)

Uwaga: W systemie AIX, Linux, and Windows IBM MQ zapewnia zgodność ze standardem FIPS 140-2 za pośrednictwem modułu szyfrującego IBM Crypto for C (ICC) . Certyfikat dla tego modułu został przeniesiony do statusu historycznego. Klienci powinni zapoznać się z informacjami w sekcji Certyfikat IBM Crypto for C (ICC) i zapoznać się z poradami NIST. Zastępczy moduł FIPS 140-3 jest obecnie w toku, a jego status można wyświetlić, wyszukując go na liście Moduły NIST CMVP na liście procesów.

Tabela 360. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSSL_FIPS_NO	0	X'00000000'
MQSSL_FIPS_YES	1	X'00000001'

MQSTAT_ * (Opcje statystyki)

Tabela 361. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSTAT_TYPE_ASYNC_ERROR	0	X'00000000'
MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)	0	X'00000000'
BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR	0	X'00000000'

MQSTS_ * (struktura struktury raportowania statusu)

Tabela 362. Struktury stałych	
Nazwa	**Struktura**
MQSTS_STRUC_ID (ID struktury MQSTS)	"STAT"
MQSTS_STRUC_ID_ARRAY	'S','T','A','T'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 363. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSTS_VERSION_1	1	X'00000001'
BIEŻĄCA_WERSJA_MQSTS	1	X'00000001'

MQSUB_ * (trwałe subskrypcje)

Trwałe dozwolone subskrypcje

Tabela 364. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSUB_DURABLE_AS_PARENT	0	X'00000000'
MQSUB_DURABLE_ALLOWED	1	X'00000001'
MQSUB_DURABLE_INHIBITED	2	X'00000002'

Zakres subskrypcji trwałych

Tabela 365. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSUB_DURABLE_ALL	-1	X'FFFFFFFF'
MQSUB_DURABLE_YES	1	X'00000001'
MQSUB_DURABLE_NO	2	X'00000002'

MQSUBTYPE_ * (typy subskrypcji w formacie komendy)

Tabela 366. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
Funkcja API MQSUBTYPE_API	1	X'00000001'
ADMINISTRATOR PODTYPU MQSUBTYPE_ADMIN	2	X'00000002'
MQSUBTYPE_PROXY	3	X'00000003'
MQSUBTYPE_ALL	-1	X'FFFFFFFF'
UŻYTKOWNIK PODTYPU MQSUBTYPE_USER	-2	X'FFFFFFFE'

MQSUS_ * (Status zawieszenia w formacie komendy)

Tabela 367. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQSUS_TAK	1	X'00000001'
MQSUS_NO	0	X'00000000'

MQSVC_ * (usługa)

Typy usług

Tabela 368. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
KOMENDA MQSVC_TYPE_COMMAND	0	X'00000000'
SERWER_MQSVC_TYPE_SERVER	1	X'00000001'

Elementy sterujące usługi

Tabela 369. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSVC_CONTROL_Q_MGR	0	X'00000000'
MQSVC_CONTROL_Q_MGR_START	1	X'00000001'
MQSVC_CONTROL_MANUAL,	2	X'00000002'

Status usługi

Tabela 370. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSVC_STATUS_ZATRZYMANY	0	X'00000000'
MQSVC_STATUS_STARTING	1	X'00000001'
MQSVC_STATUS_URUCHOMIONY	2	X'00000002'
ZATRZYMANY_STATUS_MQSVC	3	X'00000003'
MQSVC_STATUS_PONOWIENIA	4	X'00000004'

MQSYNCPOINT_ * (wartości punktu synchronizacji formatu komendy dla migracji publikowania/subskrypcji)

Tabela 371. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSYNCPOINT_YES	0	X'00000000'
MQSYNCPOINT_IFPER	1	X'00000001'

MQSYSP_ * (wartości parametrów systemowych formatu komendy)

Tabela 372. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQSYSP_NO	0	X'00000000'
MQSYSP_TAK	1	X'00000001'
MQSYSP_EXTENDED,	2	X'00000002'
MQSYSP_TYPE_INITIAL	10	X'0000000A'
MQSYSP_TYPE_SET,	11	X'0000000B'
MQSYSP_TYPE_LOG_COPY,	12	X'0000000C'
MQSYSP_TYPE_LOG_STATUS,	13	X'0000000D'

Tabela 372. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ZAJĘTY	30	X'0000001E'
MQSYSP_STATUS_PREMOUNT	31	X'0000001F'
MQSYSP_STATUS_DOSTĘPNE	32	X'00000020'
MQSYSP_STATUS_NIEZNANY	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_ * (Atrybuty tematu)

Znaki wieloznaczne

Tabela 373. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTA_BLOCK	1	X'00000001'
MQTA_PASSTHRU	2	X'00000002'

Dozwolone subskrypcje

Tabela 374. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTA_SUB_AS_PARENT	0	X'00000000'
MQTA_SUB_INHIBITED	1	X'00000001'
MQTA_SUB_ALLOWED	2	X'00000002'

Propagacja podrzędna proxy

Tabela 375. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTA_PROXY_SUB_FORCE	1	X'00000001'
MQTA_PROXY_SUB_FIRSTUSE	2	X'00000002'

Dozwolone publikacje

Tabela 376. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTA_PUB_AS_PARENT	0	X'00000000'
MQTA_PUB_INHIBITED	1	X'00000001'
MQTA_PUB_ALLOWED	2	X'00000002'

MQTC_ * (elementy sterujące wyzwalacza)

Tabela 377. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
ZMQTC_OFF	0	X'00000000'
MQTC_WŁ	1	X'00000001'

MQTCPKEEP_ * (TCP Keepalive)

Tabela 378. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTCPKEEP_NO	0	X'00000000'
MQTCPKEEP_YES	1	X'00000001'

MQTCPSTACK_ * (typy stosu TCP)

Tabela 379. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTCPSTACK_SINGLE	0	X'00000000'
MQTCPSTACK_MULTIPLE	1	X'00000001'

MQTIME_ * (format komendy-jednostki czasu)

Tabela 380. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTIME_UNIT_MINSConstellation name (optional)	0	X'00000000'
MQTIME_UNIT_SEK.	1	X'00000001'

MQTM_ * (struktura komunikatu wyzwalacza)

Tabela 381. Struktury stałych	
Nazwa	**Struktura**
MQTM_STRUC_ID	"TM¬¬"
MQTM_STRUC_ID_ARRAY	'T','M','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 382. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTM_VERSION_1	1	X'00000001'
MQTM_CURRENT_VERSION	1	X'00000001'

MQTMC_ * (struktura formatu znakowego komunikatu wyzwalacza)

Tabela 383. Struktury stałych	
Nazwa	**Struktura**
MQTMC_STRUC_ID (ID struktury MQTMC)	"TMC¬"
MQTMC_STRUC_ID_ARRAY	'T','M','C','¬'
MQTMC_VERSION_1	"¬¬¬1"

Tabela 383. Struktury stałych (kontynuacja)	
Nazwa	**Struktura**
MQTMC_VERSION_2	"¬¬¬2"
MQTMC_CURRENT_VERSION	"¬¬¬2"
MQTMC_VERSION_1_ARRAY	'¬','¬','¬','1'
MQTMC_VERSION_2_ARRAY	'¬','¬','¬','2'
MQTMC_CURRENT_VERSION_ARRAY (tablica bieżącej wersji)	'¬','¬','¬','2'

MQTOPT_ * (Typ tematu)

Tabela 384. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTOPT_LOCAL,	0	X'00000000'
KLASTER_MQTOP_CLUSTER	1	X'00000001'
MQTOPT_ALL	2	X'00000002'

MQTRAXSTR_ * (Autostart śledzenia inicjatora kanału)

Tabela 385. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTRAXSTR_NO	0	X'00000000'
MQTRAXSTR_YES	1	X'00000001'

MQTSCOPE_ * (zasięg subskrypcji)

Tabela 386. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTSCOPE_QMGR,	1	X'00000001'
MQTSCOPE_ALL	2	X'00000002'

MQTT_ * (typy wyzwalaczy)

Tabela 387. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTT_BRAK	0	X'00000000'
MQTT_FIRST	1	X'00000001'
MQTT_EVERY	2	X'00000002'
MQTT_DEPTH	3	X'00000003'

MQTYPE_ * (typy danych właściwości)

Tabela 388. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQTYPE_AS_SET	0	X'00000000'
MQTYPE_NULL	2	X'00000002'
MQTYPE_BOOLEAN (wartość boolowska)	4	X'00000004'
MQTYPE_BYTE_STRING ŁAŃCUCH	8	X'00000008'

Tabela 388. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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 (łańcuch MQTYPE)	1024	X'00000400'

MQUA_ * (selektory atrybutów użytkownika publikowania/subskrypcji)

Tabela 389. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQUA_FIRST	65536	X'00010000'
MQUA_LAST	999999999	X'3B9AC9FF'

MQUIDSUPP_ * (Obsługa ID użytkownika w formacie komendy)

Tabela 390. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQUIDSUPP_NO	0	X'00000000'
MQUIDSUPP_TAK	1	X'00000001'

MQUNDELIVERED_ * (Niedostarczone wartości formatu komendy dla migracji publikowania/subskrypcji)

Tabela 391. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQUNDELIVERED_NORMAL	0	X'00000000'
MQUNDELIVERED_SAFE	1	X'00000001'
MQUNDELIVERED_DISCARD,	2	X'00000002'
MQUNDELIVERED_KEEP	3	X'00000003'

MQUOWST_ * (stany UOW w formacie komendy)

Tabela 392. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQUOWST_BRAK	0	X'00000000'
MQUOWST_AKTYWNY	1	X'00000001'
MQUOWST_PREPARED (PREPARED)	2	X'00000002'
MQUOWST_UNRESOLVED (NIEROZWIĄZANY)	3	X'00000003'

MQUOWT_ * (Typy jednostek pracy w formacie komendy)

Tabela 393. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUOWT_Q_MGR	0	X'00000000'
MQUOWT_CICS,	1	X'00000001'
MQUOWT_RRS	2	X'00000002'
MQUOWT_IMS	3	X'00000003'
KWOWT_XA	4	X'00000004'

MQUS_ * (Użycie kolejki)

Tabela 394. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQUS_NORMAL	0	X'00000000'
MQUS_TRANSMISSION.	1	X'00000001'

MQUSAGE_ * (wartości użycia zestawu stron w formacie komendy i wartości użycia zestawu danych)

Wartości użycia zestawu stron dla formatu komendy

Tabela 395. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_PS_SUSPENDED	4	X'00000004'
MQUSAGE_EXPAND_USER	1	X'00000001'
MQUSAGE_EXPAND_SYSTEM	2	X'00000002'
MQUSAGE_EXPAND_NONE	3	X'00000003'

Wartości użycia zestawu danych w formacie komendy

Tabela 396. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (długość wartości)

Tabela 397. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQVL_NULL_TERMINATED,	-1	X'FFFFFFFF'
MQVL_EMPTY_STRING	0	X'00000000'

MQVU_ * (zmienn. ID użytk.)

Tabela 398. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQVU_FIXED_USER	1	X'00000001'
MQVU_ANY_USER	2	X'00000002'

MQWDR_ * (Struktura rekordu docelowego wyjścia obciążenia klastra)

Tabela 399. Struktury stałych	
Nazwa	**Struktura**
Identyfikator struktury MQWDR_STRUC_ID	"WDR¬"
MQWDR_STRUC_ID_ARRAY	'W','D','R','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 400. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (przedział czasu oczekiwania)

Tabela 401. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQWI_UNLIMITED	-1	X'FFFFFFFF'

MQWIH_ * (struktura i flagi nagłówka informacji o obciążeniu)

Struktura nagłówka informacji o obciążeniu

Tabela 402. Struktury stałych	
Nazwa	**Struktura**
MQWIH_ID_struktury	"WIH¬"
MQWIH_STRUC_ID_ARRAY	'W','I','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 403. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQWIH_VERSION_1	1	X'00000001'
MQWIH_CURRENT_VERSION	1	X'00000001'
MQWIH_LENGTH_1	120	X'00000078'
MQWIH_CURRENT_LENGTH	120	X'00000078'

Flagi nagłówka informacji o obciążeniu

Tabela 404. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWIH_BRAK	0	X'00000000'

MQWQR_ * (Struktura rekordu kolejki wyjścia obciążenia klastra)

Tabela 405. Struktury stałych

Nazwa	Struktura
Identyfikator QR_STRUC_ID produktu MQWQR_STRUC_ID	"WQR¬"
MQWQR_STRUC_ID_ARRAY	'W','Q','R','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 406. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
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_ * (schemat znaków wieloznacznych)

Tabela 407. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWS_DEFAULT	0	X'00000000'
MQWS_ZNAK	1	X'00000001'
MQWS_TOPIC	2	X'00000002'

MQWXP_ * (Struktura parametru wyjścia obciążenia klastra)

MQWXP_ * (Struktura parametru wyjścia obciążenia klastra)

Tabela 408. Struktury stałych

Nazwa	Struktura
Identyfikator MQWXP_STRUC_ID	"WXP¬"
MQWXP_STRUC_ID_ARRAY	'W','X','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 409. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQWXP_VERSION_1	1	X'00000001'

Tabela 409. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (flagi obciążenia klastra)

Tabela 410. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQWXP_PUT_BY_CLUSTER_CHL	2	X'00000002'

Odsyłacze pokrewne

“Pola w produkcie MQWXP -struktura parametru wyjścia obciążenia klastra” na stronie 1592
Opis pól pakietu MQWXP -struktura parametru wyjścia obciążenia klastra

MQXACT_ * (typy programu wywołującego API)

Tabela 411. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXACT_EXTERNAL	1	X'00000001'
MQXACT_INTERNAL	2	X'00000002'

MQXC_ * (komendy wyjścia)

Tabela 412. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXC_MQOPEN,	1	X'00000001'
MQXC_MQCLOSE (MQXC)	2	X'00000002'
MQXC_MQGET (MQXC)	3	X'00000003'
MQXC_MQPUT	4	X'00000004'
MQXC_MQPUT1	5	X'00000005'
MQXC_MQINQ (kolejka MQXC)	6	X'00000006'
MQXC_MQSET (zestaw MQXC)	8	X'00000008'
MQXC_MQBACK	9	X'00000009'
MQXC_MQCMIT	10	X'0000000A'

MQXCC_ * (odpowiedzi wyjścia)

Tabela 413. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXCC_OK	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'

Tabela 413. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXCC_SUPPRESS_EXIT	-5	X'FFFFFFFB'
MQXCC_CLOSE_CHANNEL,	-6	X'FFFFFFFA'
MQXCC_REQUEST_ACK	-7	X'FFFFFFF9'
MQXCC-NIEPOWODZENIE	-8	X'FFFFFFF8'

MQXDR_ * (odpowiedź wyjścia)

Tabela 414. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXDR_OK	0	X'00000000'
MQXDR_CONVERSION_FAILED.	1	X'00000001'

MQXE_ * (środowiska)

Tabela 415. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXE_INNY	0	X'00000000'
Agent MQXE_MCA	1	X'00000001'
MQXE_MCA_SVRCONN	2	X'00000002'
SERWER MQXE_COMMAND_SERVER	3	X'00000003'
MQXE_MQSC	4	X'00000004'

MQXEPO_ * (Zarejestruj strukturę opcji punktu wejścia i opcje wyjścia)

Zarejestruj strukturę opcji punktu wejścia

Tabela 416. Struktury stałych	
Nazwa	**Struktura**
MQXEPO_STRUC_ID	"XEP0"
MQXEPO_STRUC_ID_ARRAY	'X','E','P','0'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 417. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXEPO_VERSION_1	1	X'00000001'
MQXEPO_CURRENT_VERSION	1	X'00000001'

Opcje wyjścia

Tabela 418. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXEPO_BRAK	0	X'00000000'

MQXF_ * (identyfikatory funkcji API)

Tabela 419. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXF_INIT	1	X'00000001'
MQXF_TERM	2	X'00000002'
MQXF_KONN	3	X'00000003'
MQXF_CONNX	4	X'00000004'
MQXF_DYSK	5	X'00000005'
MQXF_OPEN	6	X'00000006'
MQXF_ZAMK.	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'

MQXP_ * (struktura parametru wyjścia przecięcia interfejsu API)

Tabela 420. Struktury stałych	
Nazwa	**Struktura**
MQXP_STRUC_ID (identyfikator struktury MQXC)	"XP¬¬"
MQXP_STRUC_ID_ARRAY	'X','P','¬','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 421. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXP_VERSION_1	1	X'00000001'

MQXPDA_ * (obszar określania problemu)

Tabela 422. Nazwy i wartości stałych	
Nazwa	**Wartość**
BRAK MQXPDA	X'00...00' (48 wartości pustych)
MQXPDA_NONE_ARRAY	'\0','\0',... (48 wartości pustych)

MQXPT_ * (typy transportu)

Tabela 423. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXPT_ALL	-1	X'FFFFFFFF'
MQXPT_LOCAL (lokalna MQXPT)	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_ * (struktura nagłówka kolejki transmisji)

Tabela 424. Struktury stałych	
Nazwa	**Struktura**
MQXQH_ID_STRUKTURY	"XQH¬"
MQXQH_TABLICA_ID_STRUKTURY	'X','Q','H','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 425. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQXQH_VERSION_1	1	X'00000001'
MQXQH_CURRENT_VERSION	1	X'00000001'

MQXR_ * (przyczyny wyjścia)

Tabela 426. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXR_BEFORE,	1	X'00000001'
MQXR_PO	2	X'00000002'
MQXR_CONNECTION (połączenie MQXR)	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_SEK	16	X'00000010'
MQXR_RETRY	17	X'00000011'
MQXR_AUTO_CLUSSDR	18	X'00000012'
MQXR_AUTO_RECEIVER	19	X'00000013'
MQXR_CLWL_OPEN	20	X'00000014'
MQXR_CLWL_PUT	21	X'00000015'
MQXR_CLWL_MOVE	22	X'00000016'
MQXR_CLWL_REPOS	23	X'00000017'
MQXR_CLWL_REPOS_MOVE,	24	X'00000018'
MQXR_END_BATCH	25	X'00000019'
ODEBRANO MQXR_ACK_	26	X'0000001A'
MQXR_AUTO_SVRCONN	27	X'0000001B'
MQXR_AUTO_CLUSRCVR	28	X'0000001C'
PARMS MQXR_SEC_PARMS	29	X'0000001D'

MQXR2_* (odpowiedź wyjścia 2)

Tabela 427. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXR2_PUT_WITH_DEF_ACTION	0	X'00000000'
MQXR2_PUT_WITH_DEF_USERID	1	X'00000001'
MQXR2_PUT_WITH_MSG_USERID	2	X'00000002'
MQXR2_USE_AGENT_BUFFER	0	X'00000000'
MQXR2_USE_EXIT_BUFFER	4	X'00000004'
MQXR2_DEFAULT_CONTINUATION	0	X'00000000'
MQXR2_CONTINUE_CHAIN	8	X'00000008'
MQXR2_SUPPRESS_CHAIN	16	X'00000010'
MQXR2_STATIC_CACHE	0	X'00000000'
MQXR2_DYNAMIC_CACHE	32	X'00000020'

MQXT_ * (identyfikatory wyjścia)

Tabela 428. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXT_API_CROSSING_EXIT (program zewnętrzny MQXT_API)	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_ * (wartość obszaru użytkownika programu zewnętrznego)

Tabela 429. Nazwy i wartości stałych

Nazwa	Wartość
BRAK MQXUA	X'00...00' (16 wartości pustych)
MQXUA_NONE_ARRAY	'\0','\0',... (16 wartości pustych)

MQXWD_ * (Wyjście ze struktury deskryptora oczekiwania)

Tabela 430. Struktury stałych

Nazwa	Struktura
Identyfikator struktury MQXWD_STRUC_ID	"XWD¬"
MQXWD_STRUC_ID_ARRAY	'X','W','D','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 431. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQXWD_VERSION_1	1	X'00000001'

MQZAC_ * (struktura kontekstu aplikacji)

Tabela 432. Struktury stałych

Nazwa	Struktura
MQZAC_ID_STRUKTURY	"ZAC¬"
MQZAC_STRUC_ID_ARRAY	'Z','A','C','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 433. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZAC_VERSION_1	1	X'00000001'
MQZAC_CURRENT_VERSION	1	X'00000001'

MQZAD_ * (struktura danych uprawnień)

Tabela 434. Struktury stałych	
Nazwa	**Struktura**
ID_STRUKTURY_MQZAD_STRUCT	"ZAD¬"
MQZAD_STRUC_ID_ARRAY	'Z','A','D','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 435. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQZAD_VERSION_1	1	X'00000001'
MQZAD_VERSION_2	2	X'00000002'
MQZAD_CURRENT_VERSION	2	X'00000002'

MQZAET_ * (typy jednostek usług instalowalnych)

Tabela 436. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQZAET_BRAK	0	X'00000000'
MQZAET_PRINCIPAL	1	X'00000001'
GRUPA_MQZAET	2	X'00000002'
MQZAET_UNKNOWN	3	X'00000003'

MQZAO_ * (autoryzacje usług instalowalnych)

Tabela 437. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQZAO_CONNECT	1	X'00000001'
MQZAO_BROWSE,	2	X'00000002'
MQZAO_INPUT	4	X'00000004'
MQZAO_WYNIK	8	X'00000008'
MQZAO_INQUIRE,	16	X'00000010'
MQZAO_SET	32	X'00000020'
MQZAO_PASS_IDENTITY_CONTEXT,	64	X'00000040'
MQZAO_PASS_ALL_CONTEXT (mqzao_pass_all)	128	X'00000080'
MQZAO_SET_IDENTITY_CONTEXT,	256	X'00000100'
MQZAO_SET_ALL_CONTEXT (mqzao_set_all)	512	X'00000200'
MQZAO_ALTERNATE_USER_AUTHORITY (uprawnienie użytkownika na przemian)	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_CREATE	65536	X'00010000'
MQZAO_USUŃ	131072	X'00020000'

Tabela 437. Wartości stałych (kontynuacja)		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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'
Autoryzacja MQZAO_AUTORYZACJI	8388608	X'00800000'
MQZAO_ALL_ADMIN	16646144	X'00FE0000'
MQZAO_WSZYSTKIE	16662527	X'00FE3FFF'
MQZAO_REMOVE	16777216	X'01000000'
MQZAO_BRAK	0	X'00000000'

MQZAS_ * (wersja instalowalnego interfejsu usług)

Tabela 438. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
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_ * (typy uwierzytelniania)

Tabela 439. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQZAT_INITIAL_CONTEXT,	0	X'00000000'
MQZAT_CHANGE_CONTEXT,	1	X'00000001'

MQZCI_ * (wskaźnik kontynuacji usług instalowalnych)

Tabela 440. Wartości stałych		
Nazwa	**Wartość dziesiętna**	**Wartość szesnastkowa**
MQZCI_DEFAULT	0	X'00000000'
MQZCI_CONTINUE	0	X'00000000'
MQZCI_STOP	1	X'00000001'

MQZED_ * (Struktura danych jednostki)

Tabela 441. Struktury stałych	
Nazwa	**Struktura**
MQZED_STRUC_ID (identyfikator struktury MQ)	"ZED¬"
MQZED_STRUC_ID_ARRAY	'Z','E','D','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 442. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZED_VERSION_1	1	X'00000001'
MQZED_VERSION_2	2	X'00000002'
MQZED_CURRENT_VERSION	2	X'00000002'

MQZFP_ * (struktura wolnych parametrów)

Tabela 443. Struktury stałych

Nazwa	Struktura
MQZFP_STRUC_ID (Identyfikator struktury MQZF)	"ZFP¬"
MQZFP_STRUC_ID_ARRAY	'Z','F','P','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 444. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZFP_VERSION_1	1	X'00000001'
MQZFP_CURRENT_VERSION	1	X'00000001'

MQZIC_ * (struktura kontekstu tożsamości)

Tabela 445. Struktury stałych

Nazwa	Struktura
MQZIC_STRUC_ID (Identyfikator struktury MQ)	"ZIC¬"
MQZIC_STRUC_ID_ARRAY	'Z','I','C','¬'

Uwaga: Symbol ¬ reprezentuje pojedynczy znak odstępu.

Tabela 446. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZIC_VERSION_1	1	X'00000001'
MQZIC_CURRENT_VERSION	1	X'00000001'

MQZID_ * (identyfikatory funkcji dla usług)

Identyfikatory funkcji wspólne dla wszystkich usług

Tabela 447. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
INICJOWANIE_ZMATERIALIZOWANEJ tabeli zapytania	0	X'00000000'
MQZID_TERM	1	X'00000001'

Identyfikatory funkcji dla usługi uprawnień

Tabela 448. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZID_INIT_AUTHORITY,	0	X'00000000'
MQZID_TERM_AUTHORITY	1	X'00000001'
MQZID_CHECK_AUTHORITY (uprawnienie MQZID_CHECK_)	2	X'00000002'
MQZID_COPY_ALL_AUTHORITY (uprawnienie MQZID)	3	X'00000003'
MQZID_DELETE_AUTHORITY (uprawnienia MQZID)	4	X'00000004'
MQZID_SET_AUTHORITY,	5	X'00000005'
MQZID_GET_AUTHORITY (uprawnienie MQZID)	6	X'00000006'
MQZID_GET_EXPLICIT_AUTHORITY	7	X'00000007'
MQZID_REFRESH_CACHE,	8	X'00000008'
MQZID_ENUMERATE_AUTHORITY_DATA	9	X'00000009'
MQZID_AUTHENTICATE_USER	10	X'0000000A'
MQZID_FREE_USER	11	X'0000000B'
MQZID_INQUIRE,	12	X'0000000C'
MQZID_CHECK_PRIVILEGED	13	X'0000000D'

Identyfikatory funkcji dla usługi nazw

Tabela 449. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZID_INIT_NAME	0	X'00000000'
MQZID_TERM_NAME (nazwa tabeli MQZID)	1	X'00000001'
NAZWA_WYSZUKIWANIA_MQZID	2	X'00000002'
MQZID_INSERT_NAME	3	X'00000003'
ID_MQZID_DELETE_NAME	4	X'00000004'

Identyfikatory funkcji dla usługi Userid

Tabela 450. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
ID_MQZID_INIT_USERID	0	X'00000000'
ID_MQZID_TERM_USERID	1	X'00000001'
MQZID_FIND_ID_UŻYTKOWNIKA	2	X'00000002'

MQZIO_ * (opcje inicjowania usług instalowalnych)

Tabela 451. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
Podstawowy MQZIO_PRIMARY	0	X'00000000'
MQZIO_SECONDARY	1	X'00000001'

MQZNS_ * (wersja interfejsu usługi nazw)

Tabela 452. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZNS_VERSION_1	1	X'00000001'

MQZSE_ * (wskaźnik rozpoczęcia wyliczania usług instalowalnych)

Tabela 453. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZSE_START	1	X'00000001'
MQZSE_CONTINUE	0	X'00000000'

MQZSL_ * (Wskaźnik selektora usług instalowalnych)

Tabela 454. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZSL_NIE_ZWRÓCONO	0	X'00000000'
MQZSL_XX_ENCODE_CASE_ONE zwrócona	1	X'00000001'

MQZTO_ * (opcje zakończenia usług instalowalnych)

Tabela 455. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZTO_PRIMARY	0	X'00000000'
MQZTO_SECONDARY	1	X'00000001'

MQZUS_ * (wersja interfejsu usługi ID użytkownika)

Tabela 456. Wartości stałych

Nazwa	Wartość dziesiętna	Wartość szesnastkowa
MQZUS_VERSION_1	1	X'00000001'

Typy danych używane w MQI

Informacje o typach danych, które mogą być używane w interfejsie kolejki komunikatów (Message Queue Interface-MQI). Opisy, pola i deklaracje języków dla odpowiednich języków z każdym typem danych.

Typy danych i programowanie dla interfejsu MQI

Wprowadzenie do typów danych Elementary i Structure oraz sposobu używania interfejsu MQI przez programowanie w języku C, programowanie w języku COBOL lub programowanie w języku High Level Assembler .

Podstawowe typy danych

Informacje o typach danych używanych w funkcji MQI lub funkcjach wyjścia. Zostały one szczegółowo opisane, a następnie przykłady ilustrujące sposób deklarowania podstawowych typów danych w obsługiwanych językach programowania.

Typy danych używane w funkcji MQI lub w funkcjach wyjścia są następujące:

- podstawowe typy danych lub
- Agregaty podstawowych typów danych (tablice lub struktury)

W funkcjach MQI lub w funkcjach wyjścia używane są następujące podstawowe typy danych:

Tabela 457. Podstawowe nazwy, typy i opisy typów danych

Nazwa podstawowego typu danych	Typ danych	Opis
MQBOOL	Boolowski	Typ danych MQBOOL reprezentuje wartość boolowską. Wartość 0 oznacza fałsz. Każda inna wartość reprezentuje wartość true. Obiekt MQBOOL musi być wyrównany tak, jak w przypadku typu danych MQLONG.
MQBYTE	Bajt	Typ danych MQBYTE reprezentuje jeden bajt danych. W bajcie nie jest umieszczana żadna konkretna interpretacja; jest on traktowany jako łańcuch bitów, a nie jako liczba lub znak binarny. Nie jest wymagane żadne specjalne wyrównanie. Jeśli dane MQBYTE są wysyłane między menedżerami kolejek, które używają różnych zestawów znaków lub kodowań, dane MQBYTE nie są w żaden sposób przekształcane. Pola *MsgId* i *CorrelId* w strukturze MQMD są podobne do tego. Tablica danych MQBYTE jest czasami używana do reprezentowania obszaru pamięci głównej, który nie jest znany menedżerowi kolejek. Na przykład obszar może zawierać dane komunikatu aplikacji lub strukturę. Wyrównanie granic tego obszaru musi być zgodne z charakterem zawartych w nim danych. W języku programowania C można użyć dowolnego typu danych dla parametrów funkcji, które są wyświetlane jako tablice MQBYTE. Dzieje się tak dlatego, że takie parametry są zawsze przekazywane przez adres, a w języku C parametr funkcji jest deklarowany jako wskaźnik do wartości typu void.

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQBYTEN	Łańcuch *n* bajtów	Każdy typ danych MQBYTEn reprezentuje łańcuch *n* bajtów, gdzie *n* może przyjmować jedną z następujących wartości: 8, 16, 24, 32, 40 lub 128. Każdy bajt jest opisany przez typ danych MQBYTE. Nie jest wymagane żadne specjalne wyrównanie. Jeśli dane w łańcuchu bajtowym są krótsze niż zdefiniowana długość łańcucha, dane muszą być dopełnione wartościami pustymi, aby można było wypełnić łańcuch. Gdy menedżer kolejek zwraca do aplikacji łańcuchy bajtów (na przykład w wywołaniu MQGET), menedżer kolejek jest dopełniany wartościami pustymi do zdefiniowanej długości łańcucha. Dostępne są stałe nazwane definiujące długości pól łańcuchów bajtów. Są one wymienione w sekcji "Stałe" na stronie 61
MQCHAR	Znak	Typ danych MQCHAR reprezentuje znak jednobajtowy lub jeden bajt znaku dwubajtowego lub wielobajtowego. Nie jest wymagane żadne specjalne wyrównanie. Jeśli dane MQCHAR są wysyłane między menedżerami kolejek, które używają różnych zestawów znaków lub kodowań, dane MQCHAR zwykle wymagają konwersji, aby dane zostały poprawnie zinterpretowane. Menedżer kolejek wykonuje tę czynność automatycznie dla danych MQCHAR w strukturze MQMD. Konwersja danych MQCHAR w danych komunikatu aplikacji jest sterowana przez opcję MQGMO_CONVERT określoną w wywołaniu MQGET. Więcej szczegółów zawiera opis tej opcji w sekcji "MQGMO-opcje pobierania komunikatów" na stronie 376 .

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQCHARN	Łańcuch *n* znaków	Każdy typ danych MQCHARn reprezentuje łańcuch *n* znaków, gdzie *n* może przyjmować jedną z następujących wartości: 4, 8, 12, 20, 28, 32, 48, 64, 128 lub 256. Każdy znak jest opisywany przez typ danych MQCHAR. Nie jest wymagane żadne specjalne wyrównanie. Jeśli dane w łańcuchu są krótsze niż zdefiniowana długość łańcucha, dane muszą być uzupełnione odstępami, aby można było wypełnić łańcuch. W niektórych przypadkach do przedwczesnego zakończenia łańcucha zamiast dopełniania odstępami można użyć znaku o kodzie zero; znak o kodzie zero i następujące po nim znaki są traktowane jako odstępy aż do zdefiniowanej długości łańcucha. Miejsca, w których można używać wartości NULL, są identyfikowane w opisach wywołania i typu danych. Gdy menedżer kolejek zwraca łańcuchy znaków do aplikacji (na przykład w wywołaniu MQGET), menedżer kolejek zawsze dopełnia znakami spacji do zdefiniowanej długości łańcucha. Menedżer kolejek nie używa znaku o kodzie zero do rozgraniczenia łańcucha. Dostępne są stałe nazwane, które definiują długości pól łańcucha znaków i są wymienione w sekcji “Stałe” na stronie 61.

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQFLOAT32	32-bitowa liczba zmiennopozycyjna	Typ danych MQFLOAT32 to 32-bitowa liczba zmiennopozycyjna reprezentowana przy użyciu standardowego formatu zmiennopozycyjnego IEEE. Obiekt MQFLOAT32 musi być wyrównany do granicy 4-bajtowej. Użycie interfejsu MQFLOAT32 w języku C w systemie z/OS wymaga użycia flagi kompilatora FLOAT (IEEE). Użycie interfejsu MQFLOAT32 w języku COBOL jest ograniczone do kompilatorów obsługujących liczby zmiennopozycyjne w formacie IEEE. Może to wymagać użycia flagi kompilatora FLOAT (NATIVE).
MQFLOAT64	64-bitowa liczba zmiennopozycyjna	Typ danych MQFLOAT64 to 64-bitowa liczba zmiennopozycyjna reprezentowana przy użyciu standardowego formatu zmiennopozycyjnego IEEE. Wartość MQFLOAT64 musi być wyrównana do granicy 8-bajtowej. Użycie interfejsu MQFLOAT64 w języku C w systemie z/OS wymaga użycia flagi kompilatora FLOAT (IEEE). Użycie interfejsu MQFLOAT64 w języku COBOL jest ograniczone do kompilatorów obsługujących liczby zmiennopozycyjne w formacie IEEE. Może to wymagać użycia flagi kompilatora FLOAT (NATIVE).

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQHCONFIG (konfiguracja MQ)	Uchwyt konfiguracji	Typ danych MQHCONFIG reprezentuje uchwyt konfiguracji, czyli komponent, który jest konfigurowany dla konkretnej instalowalnej usługi. Uchwyt konfiguracji musi być wyrównany do swojej granicy naturalnej. Aplikacje nie mogą polegać na formacie danych przechowywanych w tym uchwycie. Jeśli jest poprawna, jej wartość jest przeznaczona do użycia w kolejnych wywołaniach MQI, ale nie ma żadnego znaczenia poza tym celem.
MQHCONN	Uchwyt połączenia	Typ danych MQHCONN reprezentuje uchwyt połączenia, czyli połączenie z konkretnym menedżerem kolejek. Uchwyt połączenia musi być wyrównany do granicy 4-bajtowej. Aplikacje nie mogą polegać na formacie danych przechowywanych w tym uchwycie. Jeśli jest poprawna, jej wartość jest przeznaczona do użycia w kolejnych wywołaniach MQI, ale nie ma żadnego znaczenia poza tym celem.
MQHMSG	uchwyt komunikatu	Typ danych MQHMSG reprezentuje uchwyt komunikatu, który zapewnia dostęp do komunikatu. Uchwyt komunikatu musi być wyrównany do 8-bajtowej granicy. Aplikacje nie mogą polegać na formacie danych przechowywanych w tym uchwycie. Jeśli jest poprawna, jej wartość jest przeznaczona do użycia w kolejnych wywołaniach MQI, ale nie ma żadnego znaczenia poza tym celem.

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQHOBJ	Uchwyt obiektu	Typ danych MQHOBJ reprezentuje uchwyt obiektu, który daje dostęp do obiektu. Uchwyt obiektu musi być wyrównany do granicy 4-bajtowej. Aplikacje nie mogą polegać na formacie danych przechowywanych w tym uchwycie. Jeśli jest poprawna, jej wartość jest przeznaczona do użycia w kolejnych wywołaniach MQI, ale nie ma żadnego znaczenia poza tym celem.
MQINT8	8-bitowa liczba całkowita ze znakiem	Typ danych MQINT8 to 8-bitowa liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -128 do +127, chyba że kontekst ogranicza inaczej.
MQINT16	16-bitowa liczba całkowita ze znakiem	Typ danych MQINT16 to 16-bitowa liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -32 768 do +32 767, chyba że kontekst stanowi inaczej. Obiekt MQINT16 musi być wyrównany do granicy dwubajtowej.
MQINT32	32-bitowa liczba całkowita ze znakiem	Typ danych MQINT32 to 32-bitowa binarna liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -2 147 483 648 do + 2 147 483 647, o ile kontekst nie ogranicza inaczej. Patrz definicja MQLONG.
MQINT64	64-bitowa liczba całkowita ze znakiem	Typ danych MQINT64 to 64-bitowa liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, o ile nie jest to ograniczone przez kontekst. Dla języka COBOL poprawny zakres jest ograniczony do -999 999 999 999 999 999 do +999 999 999 999 999 999. 64-bitowa liczba całkowita musi być wyrównana do granicy 8-bajtowej.

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQLONG	32-bitowa liczba całkowita ze znakiem	Typ danych MQLONG jest 32-bitową binarną liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -2 147 483 648 do + 2 147 483 647, chyba że kontekst ogranicza inaczej. Dla języka COBOL poprawny zakres jest ograniczony do zakresu od -999 999 999 do +999 999 999. Wartość MQLONG musi być wyrównana do granicy 4-bajtowej.
Identyfikator MQPID	Identyfikator procesu	Identyfikator procesu IBM MQ . Jest to ten sam identyfikator, który jest używany w zrzutach MQ i FFST™ , ale może być inny niż identyfikator procesu systemu operacyjnego.
MQPTR	Pointer	Typ danych MQPTR to adres danych dowolnego typu. Wskaźnik musi być wyrównany do swojej granicy naturalnej; jest to granica 16-bajtowa w systemie IBM ii granica 8-bajtowa na innych platformach. Niektóre języki programowania obsługują wskaźniki określonego typu; MQI używa ich również w kilku przypadkach (na przykład PMQCHAR i PMQLONG w języku programowania C).
ID_zmaterializowanej tabeli zapytania	Identyfikator wątku	Identyfikator wątku IBM MQ . Jest to ten sam identyfikator, który jest używany w śledzeniu MQ i zrzutach FFST™ , ale może być inny niż identyfikator wątku systemu operacyjnego.
MQUINT8	8-bitowa liczba całkowita bez znaku	Typ danych MQUINT8 to 8-bitowa liczba całkowita bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +255, o ile nie jest to ograniczone przez kontekst.

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
MQUINT16	16-bitowa liczba całkowita bez znaku	Typ danych MQUINT16 to 16-bitowa liczba całkowita bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +65 535, chyba że kontekst ogranicza inaczej. Wartość MQUINT16 musi być wyrównana do granicy dwubajtowej.
MQUINT32	32-bitowa liczba całkowita bez znaku	Typ danych MQUINT32 to 32-bitowa binarna liczba całkowita bez znaku. Patrz definicja MQULONG.
MQUINT64	64-bitowa liczba całkowita bez znaku	Typ danych MQINT64 to 64-bitowa liczba całkowita bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +18 446 744 073 709 551 615, o ile kontekst nie ogranicza się inaczej. Dla języka COBOL poprawny zakres jest ograniczony do zakresu od 0 do +999 999 999 999 999. 64-bitowa liczba całkowita musi być wyrównana do granicy 8-bajtowej.
MQULONG	32-bitowa liczba całkowita bez znaku	Typ danych MQULONG jest 32-bitową binarną liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do + 4 294 967 294, chyba że kontekst ogranicza inaczej. Dla języka COBOL poprawny zakres jest ograniczony do zakresu od 0 do +999 999 999. Obiekt MQULONG musi być wyrównany do granicy 4-bajtowej.
PMQACH	Pointer	Wskaźnik do struktury danych typu MQACH
PMQAIR	Pointer	Wskaźnik do struktury danych typu MQAIR
PMQAXC	Pointer	Wskaźnik do struktury danych typu MQAXC
PMQAXP	Pointer	Wskaźnik do struktury danych typu MQAXP
PMQBMHO	Pointer	Wskaźnik do struktury danych typu MQBMHO

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
PMQBO	Pointer	Wskaźnik do struktury danych typu MQBO
PMQBOOL	Pointer	Wskaźnik do danych typu MQBOOL
PMQBYTE,	Pointer	Wskaźnik do danych typu MQBYTE
PMQBYTEN	Pointer	Wskaźnik do danych typu MQBYTEn, gdzie n może mieć wartość 8, 16, 24, 32, 40, 128
PMQCBC,	Pointer	Wskaźnik do struktury danych typu MQCBC
PMQCBD,	Pointer	Wskaźnik do struktury danych typu MQCBD
PMQCHAR	Pointer	Wskaźnik do danych typu MQCHAR
PMQCHARN	Pointer	Wskaźnik do typu danych MQCHARN, gdzie n może mieć wartość 4, 8, 12, 20, 28, 32, 48, 64, 128, 256, 264
PMQCHARV	Pointer	Wskaźnik do struktury danych typu MQCHARV
PMQCIH	Pointer	Wskaźnik do struktury danych typu MQCIH
PMQCMHO	Pointer	Wskaźnik do struktury danych typu MQCMHO
PMQCNO	Pointer	Wskaźnik do struktury danych typu MQCNO
Protokół PMQCSP	Pointer	Wskaźnik do struktury danych typu MQCSP
PMQCTLO	Pointer	Wskaźnik do struktury danych typu MQCTLO
PMQDH	Pointer	Wskaźnik do struktury danych typu MQDH
PMQDHO	Pointer	Wskaźnik do struktury danych typu MQDHO
PMQDLH	Pointer	Wskaźnik do struktury danych typu MQDLH
PMQDMHO	Pointer	Wskaźnik do struktury danych typu MQDMHO
PMQDMPO	Pointer	Wskaźnik do struktury danych typu MQDMPO
PMQEPH	Pointer	Wskaźnik do struktury danych typu MQEPH

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
PMQFLOAT32	Pointer	Wskaźnik do struktury danych typu MQFLOAT32
PMQFLOAT64	Pointer	Wskaźnik do struktury danych typu MQFLOAT64
PMQFUNC	Pointer	Wskaźnik do funkcji
PMQGMO	Pointer	Wskaźnik do struktury danych typu MQGMO
Komenda PMQHCONFIG	Pointer	Wskaźnik do danych typu MQHCONFIG
PMQHCONN	Pointer	Wskaźnik do danych typu MQHCONN
PMQHMSG	Pointer	Wskaźnik do danych typu MQHMSG
PMQHOBJ	Pointer	Wskaźnik do danych typu MQHOBJ
PMQIIH	Pointer	Wskaźnik do struktury danych typu MQIIH
PMQIMPO	Pointer	Wskaźnik do struktury danych typu MQIMPO
PMQINT8	Pointer	Wskaźnik do danych typu MQINT8
PMQINT16	Pointer	Wskaźnik do danych typu MQINT16
PMQINT32	Pointer	Wskaźnik do danych typu MQINT32
PMQINT64	Pointer	Wskaźnik do danych typu MQINT64
PMQLONG	Pointer	Wskaźnik do danych typu MQLONG
PMQMD	Pointer	Wskaźnik do struktury typu MQMD
PMQMDE,	Pointer	Wskaźnik do struktury danych typu MQMDE
PMQMD1	Pointer	Wskaźnik do struktury danych typu MQMD1
PMQMD2	Pointer	Wskaźnik do struktury danych typu MQMD2
PMQMHBO,	Pointer	Wskaźnik do struktury danych typu MQMHBO
PMQOD	Pointer	Wskaźnik do struktury danych typu MQOD
PMQOR,	Pointer	Wskaźnik do struktury danych typu MQOR
Komenda PMQPD	Pointer	Wskaźnik do struktury danych typu MQPD
PMQPID	Pointer	Wskaźnik do identyfikatora procesu

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)

Nazwa podstawowego typu danych	Typ danych	Opis
PMQMD	Pointer	Wskaźnik do struktury danych typu MQMD
PMQPMO	Pointer	Wskaźnik do struktury danych typu MQPMO
PMQPTR	Pointer	Wskaźnik do danych typu MQPTR
PMQRFH	Pointer	Wskaźnik do struktury danych typu MQRFH
PMQRFH2	Pointer	Wskaźnik do struktury danych typu MQRFH2
PMQRMH	Pointer	Wskaźnik do struktury danych typu MQRMH
PMQRR	Pointer	Wskaźnik do struktury danych typu MQRR
PMQSCO	Pointer	Wskaźnik do struktury danych typu MQSCO
PMQSD	Pointer	Wskaźnik do struktury danych typu MQSD
PMQSMPO	Pointer	Wskaźnik do struktury danych typu MQSMPO
PMQSRO	Pointer	Wskaźnik do struktury danych typu MQSRO
PMSSTS	Pointer	Wskaźnik do struktury danych typu MQSTS
Identyfikator PMQTID	Pointer	Wskaźnik do identyfikatora wątku
PMQTM,	Pointer	Wskaźnik do struktury danych typu MQTM
PMQTMC2	Pointer	Wskaźnik do struktury danych typu MQTMC2
PMQUINT8	Pointer	Wskaźnik do typu danych MQUINT8
PMQUINT16	Pointer	Wskaźnik do typu danych MQUINT16
PMQUINT32	Pointer	Wskaźnik do typu danych MQUINT32
PMQUINT64	Pointer	Wskaźnik do typu danych MQUINT64
PMQULONG	Pointer	Wskaźnik do typu danych MQULONG
PMQVOID (identyfikator produktu MQ)	Pointer	
PMQWIH	Pointer	Wskaźnik do struktury danych typu MQWIH

Tabela 457. Podstawowe nazwy, typy i opisy typów danych (kontynuacja)		
Nazwa podstawowego typu danych	**Typ danych**	**Opis**
PMQXQH	Pointer	Wskaźnik do struktury danych typu MQXQH

Deklaracje typu danych C

Tabela 458. Nazwy i reprezentacje typów danych języka C	
Typ danych	**Reprezentacja**
MQBOOL	`typedef MQLONG MQBOOL;`
MQBYTE	`typedef unsigned char MQBYTE;`
MQBYTE8	`typedef MQBYTE MQBYTE8[8];`
MQBYTE16	`typedef MQBYTE MQBYTE16[16];`
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];`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)

Typ danych	**Reprezentacja**
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 (konfiguracja MQ)	`typedef void MQPOINTER MQHCONFIG;`
MQHCONN	`typedef MQLONG MQHCONN;`
MQHOBJ	`typedef MQLONG MQHOBJ;`
MQINT8	`typedef signed char MQINT8;`
MQINT16	`typedef short MQINT16;`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)

Typ danych	**Reprezentacja**

MQINT64	UNIX W 64-bitowym systemie UNIX:
	`typedef long;`
	AIX W 32-bitowych systemach AIX:
	`typedef int64_t;`
	IBM i Linux z/OS W systemach Linux, IBM ii z/OS:
	`typedef long long;`
	Windows W systemie Windows:
	`typedef _int64;`

MQLONG	IBM i W systemie IBM i:
	`typedef long MQLONG;`
	z/OS ALW Na innych platformach:
	`if defined(MQ_64_BIT)`
	`  typedef int MQLONG;`
	`else`
	`  typedef long MQLONG;`

Identyfikator MQPID	`typedef MQLONG MQPID;`
MQPTR	`typedef void MQPOINTER MQPTR;`
ID_zmaterializowanej tabeli zapytania	`typedef MQLONG MQTID;`
MQUINT8	`typedef unsigned char MQUINT8;`
MQUINT16	`typedef unsigned short MQUINT16;`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)

Typ danych	Reprezentacja

MQUINT64	UNIX W 64-bitowym systemie UNIX:
	`typedef unsigned long;`
	AIX W 32-bitowych systemach AIX:
	`typedef uint64_t;`
	IBM i Linux z/OS W systemach Linux, IBM ii z/OS:
	`typedef unsigned long long;`
	Windows W systemie Windows:
	`typedef unsigned _int64;`

MQULONG	IBM i W systemie IBM i:
	`typedef unsigned long MQULONG;`
	z/OS ALW Na innych platformach:
	`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];`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)

Typ danych	Reprezentacja
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];`
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;`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)

Typ danych	Reprezentacja
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;`
Komenda 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;`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)

Typ danych	Reprezentacja
PMQOD	`typedef MQOD MQPOINTER PMQOD;`
PMQPMO	`typedef MQPMO MQPOINTER PMQPMO;`
PMQPTR	`typedef MQPTR MQPOINTER PMQPTR;`
PMQRFH	`typedef MQRFH MQPOINTER PMQRFH;`
PMQRFH2	`typedef MQRFH2[2] MQPOINTER PMQRFH2[2];`
PMQRMH	`typedef MQRMH MQPOINTER PMQRMH;`
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 (identyfikator produktu MQ)	`typedef void MQPOINTER PMQVOID;`
PMQWIH	`typedef MQWIH MQPOINTER PMQWIH;`
PMQXQH	`typedef MQXQH MQPOINTER PMQXQH;`
PPMQBO	`typedef PMQBO MQPOINTER PPMQBO;`
PPMQBAJT	`typedef PMQBYTE MQPOINTER PPMQBYTE;`

Tabela 458. Nazwy i reprezentacje typów danych języka C (kontynuacja)	
Typ danych	**Reprezentacja**
PPMQCHAR	`typedef PMQCHAR MQPOINTER PPMQCHAR;`
PPMQCNO	`typedef PMQCNO MQPOINTER PPMQCNO;`
PPMQGMO	`typedef PMQGMO MQPOINTER PPMQGMO;`
PPMQHCONN	`typedef PMQHCONN MQPOINTER PPMQHCONN;`
PPMQHOBJ	`typedef PMQHOBJ MQPOINTER PPMQHOBJ;`
PPMQLONG	`typedef PMQLONG MQPOINTER PPMQLONG;`
PPMQMD (PMQMD)	`typedef PMQMD MQPOINTER PPMQMD;`
PPMQOD	`typedef PMQOD MQPOINTER PPMQOD;`
PPMQPMO	`typedef PMQPMO MQPOINTER PPMQPMO;`
PPMQULONG	`typedef PMQULONG MQPOINTER PPMQULONG;`
PPMQVOID	`typedef PMQVOID MQPOINTER PPMQVOID;`
Gdzie `defined(MQ_64_BIT)` oznacza platformę 64-bitową.	

Opis zmiennej makra MQPOINTER można znaleźć w sekcji “Typy danych” na stronie 266 .

Deklaracje typów danych COBOL

Tabela 459. Nazwy i reprezentacje typów danych w języku COBOL	
Typ danych	**Reprezentacja**
MQBOOL	`PIC S9(9) BINARY`
MQBYTE	`PIC X`

Tabela 459. Nazwy i reprezentacje typów danych w języku COBOL (kontynuacja)

Typ danych	**Reprezentacja**
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)`
MQCHAR32	`PIC X(32)`
MQCHAR48	`PIC X(48)`
MQCHAR64	`PIC X(64)`
MQCHAR128	`PIC X(128)`
MQCHAR256	`PIC X(256)`

Tabela 459. Nazwy i reprezentacje typów danych w języku COBOL (kontynuacja)

Typ danych	**Reprezentacja**
MQFLOAT32	`USAGE COMP-1`
MQFLOAT64	`USAGE COMP-2`

MQHCONN	wł.z/OS
	`PIC S9(9) COMP-5`
	Na innych platformach
	`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`

Deklaracje typów danych języka PL/I

Tabela 460. Nazwy i reprezentacje typów danych języka PL/I	
Typ danych	**Reprezentacja**
MQBOOL	`fixed bin(31)`
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)`

Tabela 460. Nazwy i reprezentacje typów danych języka PL/I (kontynuacja)

Typ danych	Reprezentacja
MQCHAR128	`char(128)`
MQCHAR256	`char(256)`
MQFLOAT32	`binary float(21) ieee`
MQFLOAT64	`binary float(52) ieee`
MQHCONN	`fixed bin(31)`
MQHOBJ	`fixed bin(31)`
MQINT8	`fixed bin(7)`
MQINT16	`fixed bin(15)`
MQINT64	`fixed bin(63)`
MQLONG	`fixed bin(31)`
MQPTR	`pointer`
MQUINT8	`fixed bin(8)`
MQUINT16	`fixed bin(16)`
MQUINT64	`fixed bin(64)`
MQULONG	`fixed bin(32)`

Deklaracje typów danych High Level Assembler

Tabela 461. Nazwy i reprezentacje typów danych asemblera System/390	
Typ danych	**Reprezentacja**
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`
MQCHAR12	`DS CL12`
MQCHAR20	`DS CL20`
MQCHAR28	`DS CL28`
MQCHAR32	`DS CL32`
MQCHAR48	`DS CL48`
MQCHAR64	`DS CL64`

Tabela 461. Nazwy i reprezentacje typów danych asemblera System/390 (kontynuacja)	
Typ danych	**Reprezentacja**
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`

Typy danych struktury

Podsumowanie typów danych struktury, reguł spójnego odwzorowywania struktur MQI i konwencji używanych w każdym opisie typu danych struktury.

- “Podsumowanie typów danych struktury używanych w wywołaniach MQI lub funkcjach wyjścia” na stronie 262
- “Podsumowanie typów danych struktury używanych w danych komunikatu” na stronie 263
- “Reguły spójnego odwzorowywania struktur MQI” na stronie 263
- “Konwencje używane w każdym opisie typu danych struktury” na stronie 264

Podsumowanie typów danych struktury używanych w wywołaniach MQI lub funkcjach wyjścia

Tabela 462. Typy danych struktury używane w wywołaniach MQI lub funkcjach wyjścia

Struktura	Opis	Wywołania tam, gdzie są używane
ZMQACH	Nagłówek łańcucha wyjścia funkcji API	
MQAIR	Rekord informacji uwierzytelniającej	MQCONNX (usługa MQCONNX)
MQAXC	Kontekst wyjścia funkcji API	
MQAXP	Parametr wyjścia funkcji API	
MQBMHO	Opcje uchwytu buforu do komunikatu	MQBUFMH
MQBO	Opcje początku	MQBEGIN
MQCBD	Deskryptor wywołania zwrotnego	Baza MQCB
MQCBO	Opcje tworzenia wielozbioru	mqCreate-wielozbiór
MQCHARV	Łańcuch o zmiennej długości	MQINQMP
MQCNO	Opcje połączenia	MQCONNX (usługa MQCONNX)
MQCSP (MQCSP)	Parametry zabezpieczeń	MQCONNX (usługa MQCONNX)
MQCTLO	Opcje wywołania zwrotnego	MQCTL (MQCTL)
MQDMPO	Opcje usuwania właściwości komunikatu	MQDLTMP
MQGMO (MQGMO)	Opcje pobierania komunikatów	MQGet
MQIMPO	Sprawdź opcje właściwości komunikatu	MQINQMP
MQMD	deskryptor komunikatu	MQBUFMH, MQMHBUF, MQCB, MQGET, MQPUT, MQPUT1
MQMHBO (interfejs MQMHBO)	Opcje uchwytu komunikatu do buforu	MQMHBUF
MQOD	deskryptor obiektu	MQOPEN, MQPUT1
MQOR	Rekord obiektu	MQOPEN, MQPUT1
MQPD	Deskryptor właściwości	MQSETMP (komenda MQSETMP)
MQPMO	Opcje umieszczania komunikatów	MQPUT, MQPUT1
Raport MQPMR	Put-message record (rekord komunikatu umieszczania)	MQPUT, MQPUT1
MQRR (MQRR)	Rekord odpowiedzi	MQOPEN, MQPUT, MQPUT1

Tabela 462. Typy danych struktury używane w wywołaniach MQI lub funkcjach wyjścia (kontynuacja)		
Struktura	**Opis**	**Wywołania tam, gdzie są używane**
MQSCO (usługa MQSCO)	Opcje konfiguracyjne TLS	MQCONNX (usługa MQCONNX)
Tabela MQSD	Deskryptor subskrypcji	MQSUB (MQSUB)
MQSMPO	Ustaw opcję właściwości komunikatu	MQSETMP (komenda MQSETMP)
MQSRO	Opcje żądania subskrypcji	MQSUBRQ (MQSUBRQ)
MQSTS	Struktura raportowania statusu	MQSTAT (tabela MQSTAT)

Podsumowanie typów danych struktury używanych w danych komunikatu

Tabela 463. Typy danych struktury używane w danych komunikatu	
Struktura	**Opis**
MQCIH,	CICS nagłówek informacji
MQCFH	Nagłówek PCF
MQEPH	Osadzony nagłówek PCF
MQDH	Nagłówek dystrybucji
MQDLH	Nagłówek niedostarczonego komunikatu
MQIIH.	IMS nagłówek informacji
MQMDE	Rozszerzenie deskryptora komunikatu
MQRFH,	Reguły i nagłówek formatowania
MQRFH2	Reguły i formatowanie nagłówka 2
MQRMH	Nagłówek komunikatu referencyjnego
MQTM	komunikat wyzwalacza
MQTMC2	Komunikat wyzwalacza (format znakowy 2)
MQWIH	Nagłówek informacji o pracy
MQXQH	Nagłówek kolejki transmisji

Uwaga: Struktura MQDXP (parametr wyjścia konwersji danych) jest opisana w sekcji “Wyjście konwersji danych” na stronie 937wraz z powiązanymi wywołaniami konwersji danych.

Reguły spójnego odwzorowywania struktur MQI

Języki programowania różnią się poziomem obsługi struktur, a niektóre reguły i konwencje są stosowane w celu spójnego odwzorowania struktur MQI w każdym języku programowania:

1. Struktury muszą być dopasowane do ich naturalnych granic.
   - Większość struktur MQI wymaga wyrównania 4-bajtowego.
   - W systemie IBM istruktury zawierające wskaźniki wymagają 16-bajtowego wyrównania. Są to: MQCNO, MQOD, MQPMO.
2. Każde pole w strukturze musi być wyrównane względem swojej granicy naturalnej.
   - Pola z typami danych równymi z MQLONG muszą być wyrównane do 4-bajtowych granic.

- Pola z typami danych równymi MQPTR muszą być wyrównane do 16-bajtowych granic w systemie IBM ii do 4-bajtowych granic w innych środowiskach.
- Inne pola są wyrównywane do granic 1-bajtowych.

3. Długość struktury musi być wielokrotnością jej wyrównania granicy.
   - Większość struktur MQI ma długości, które są wielokrotnościami 4 bajtów.
   - W systemie IBM istruktury zawierające wskaźniki mają długość będącą wielokrotnością 16 bajtów.
4. W razie potrzeby należy dodać bajty lub pola dopełniające, aby zapewnić zgodność z poprzednimi regułami.

Konwencje używane w każdym opisie typu danych struktury

Opis każdego typu danych struktury zawiera:

- Przegląd celu i zastosowania struktury
- Opisy pól w strukturze, w formie niezależnej od języka programowania
- Przykłady sposobu deklarowania struktury w każdym z obsługiwanych języków programowania

Opis każdego typu danych struktury zawiera następujące sekcje:

Nazwa struktury
Nazwa struktury, po której następuje podsumowanie pól w strukturze.

Przegląd
Krótki opis przeznaczenia i wykorzystania struktury.

Pola
Opisy pól. Dla każdego pola po nazwie pola występuje podstawowy typ danych w nawiasach (). W tekście nazwy pól są wyświetlane przy użyciu kursywy, na przykład *Version*.

Dostępny jest również opis przeznaczenia pola wraz z listą wartości, które mogą być użyte w tym polu. Nazwy stałych są wyświetlane wielkimi literami, na przykład MQGMO_STRUC_ID. Zestaw stałych o takim samym przedrostku jest wyświetlany przy użyciu znaku *, na przykład: MQIA_ *.

W opisach pól używane są następujące terminy:

dane wejściowe
Informacje są podawane w polu podczas wykonywania połączenia.

wyniki
Menedżer kolejek zwraca informacje w polu po zakończeniu lub niepowodzeniu wywołania.

Wejście/wyjście
Informacje są podawane w polu podczas wykonywania wywołania, a menedżer kolejek zmienia informacje po zakończeniu lub niepowodzeniu wywołania.

Wartości początkowe
Tabela przedstawiająca wartości początkowe dla każdego pola w plikach definicji danych dostarczonych z produktem MQI.

Deklaracja C
Typowa deklaracja konstrukcji w C.

Deklaracja języka COBOL
Typowa deklaracja struktury w języku COBOL.

Deklaracja PL/I
Typowa deklaracja konstrukcji w PL/I.

Deklaracja High Level Assembler
Typowa deklaracja struktury w języku asemblera System/390 .

Deklaracja Visual Basic
Typowa deklaracja konstrukcji w Visual Basic.

Programowanie w języku C

Informacje pomocne w korzystaniu z interfejsu MQI z języka programowania C.

Pliki nagłówkowe

Tabela 464. Pliki nagłówkowe języka C

Plik	Spis treści
CMQC	Prototypy funkcji, typy danych i stałe nazwane dla głównego interfejsu MQI
CMQXC	Prototypy funkcji, typy danych i stałe nazwane dla wyjścia konwersji danych
CMQEC	Prototypy funkcji, typy danych i stałe nazwane dla głównego interfejsu MQI, wyjścia konwersji danych i struktury punktów wejścia interfejsu (CMQEC zawiera CMQXC i CMQC).
Komenda CMQSTRC	Funkcje, które przekształcają definicje stałych MQI w odpowiedniki tekstowe. **Ostrzeżenie:** z/OS Dotyczy z/OS z IBM MQ 9.1. Programy używające tego pliku nagłówkowego muszą być skompilowane z opcją kompilatora LONGNAME.

Aby poprawić przenośność aplikacji, należy zakodować nazwę pliku nagłówkowego małymi literami w dyrektywie preprocesora `#include` :

```
#include "cmqec.h"
```

Funkcje

Nie trzeba podawać wszystkich parametrów, które są przekazywane przez adres przy każdym wywołaniu funkcji.

- Przekaż parametry, które są *tylko wejściowe* i mają typ MQHCONN, MQHOBJ lub MQLONG według wartości.
- Przekaż wszystkie inne parametry według adresu.

Jeśli konkretny parametr nie jest wymagany, należy użyć pustego wskaźnika jako parametru w wywołaniu funkcji zamiast adresu danych parametru. Parametry, dla których jest to możliwe, są określone w opisach wywołań.

Żaden parametr nie jest zwracany jako wartość funkcji; w terminologii C oznacza to, że wszystkie funkcje zwracają wartość `void`.

Atrybuty funkcji są definiowane przez zmienną makra MQENTRY. Wartość tej zmiennej makra zależy od środowiska.

Parametry z niezdefiniowanym typem danych

Parametr **Buffer** w funkcjach MQGET, MQPUT i MQPUT1 ma niezdefiniowany typ danych. Ten parametr służy do wysyłania i odbierania danych komunikatu aplikacji.

Parametry tego sortowania są wyświetlane w przykładach w języku C jako tablice MQBYTE. W ten sposób można zadeklarować parametry, ale zwykle wygodniej jest zadeklarować je jako konkretną strukturę, która opisuje układ danych w komunikacie. Zadeklaruj rzeczywisty parametr funkcji jako wskaźnik do unieważnienia i określ adres dowolnego rodzaju danych jako parametr w wywołaniu funkcji.

Typy danych

Zdefiniuj wszystkie typy danych za pomocą instrukcji `typedef` języka C. Dla każdego typu danych należy również zdefiniować odpowiedni typ danych wskaźnika. Nazwa typu danych wskaźnika jest nazwą typu danych elementarnych lub strukturalnych z przedrostkiem P oznaczającym wskaźnik. Zdefiniuj atrybuty wskaźnika przy użyciu zmiennej makra MQPOINTER. Wartość tej zmiennej makra zależy od środowiska. Poniżej przedstawiono sposób deklarowania typów danych wskaźnika:

```
#define MQPOINTER *                /* depends on environment */
...
typedef MQLONG MQPOINTER PMQLONG;  /* pointer to MQLONG */
typedef MQMD   MQPOINTER PMQMD;    /* pointer to MQMD */
```

Manipulowanie łańcuchami binarnymi

Zadeklaruj łańcuchy danych binarnych jako jeden z typów danych MQBYTEn.

Podczas kopiowania, porównywania lub ustawiania pól tego typu należy używać funkcji języka C **memcpy**, **memcmp**lub **memset** ; na przykład:

```
#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));
```

Nie należy używać funkcji łańcuchowych **strcpy**, **strcmp**, **strncpy**lub **strncmp**, ponieważ nie działają one poprawnie dla danych zadeklarowanych z typami danych MQBYTEn.

Manipulowanie łańcuchami znaków

Gdy menedżer kolejek zwraca dane znakowe do aplikacji, dane znakowe są zawsze dopełniane odstępami do zdefiniowanej długości pola. Menedżer kolejek *nie* zwraca łańcuchów zakończonych znakiem o kodzie zero.

Dlatego podczas kopiowania, porównywania lub konkatenowania takich łańcuchów należy używać funkcji łańcuchowych **strncpy**, **strncmp**lub **strncat**.

Nie należy używać funkcji łańcuchowych, które wymagają, aby łańcuch był zakończony wartością NULL (**strcpy**, **strcmp**, **strcat**). Ponadto nie należy używać funkcji **strlen** do określenia długości łańcucha. Zamiast niej należy użyć funkcji **sizeof** do określenia długości pola.

Wartości początkowe dla struktur

Pliki nagłówkowe definiują różne zmienne makra, których można użyć do udostępnienia wartości początkowych dla struktur MQ podczas deklarowania instancji tych struktur.

Te zmienne makra mają nazwy w postaci MQxxx_DEFAULT, gdzie MQxxx reprezentuje nazwę struktury. Są one używane w następujący sposób:

```
MQMD   MyMsgDesc = {MQMD_DEFAULT};
MQPMO  MyPutOpts = {MQPMO_DEFAULT};
```

W przypadku niektórych pól znakowych (na przykład pól *StrucId* , które występują w większości struktur, lub pola *Format* , które występuje w strukturze MQMD) interfejs MQI definiuje konkretne poprawne wartości. Dla każdej z poprawnych wartości podano *dwie* zmienne makra:

- Jedna zmienna makra definiuje wartość jako łańcuch o długości, z wyłączeniem niejawnych dopasowań wartości NULL, dokładnie zdefiniowanej długości pola. Na przykład dla pola *Format* w strukturze MQMD udostępniono następującą zmienną makra (¬ reprezentuje pojedynczy znak odstępu):

  ```
  #define MQFMT_STRING "MQSTR¬¬¬"
  ```

  Tego formularza należy używać z funkcjami `memcpy` i `memcmp` .
- Inna zmienna makra definiuje wartość jako tablicę znaków; nazwa tej zmiennej makra jest nazwą łańcucha z przyrostkiem _ARRAY. Na przykład:

  ```
  #define MQFMT_STRING_ARRAY 'M','Q','S','T','R','¬','¬','¬'
  ```

  Ten formularz służy do inicjowania pola podczas deklarowania instancji struktury z wartościami innymi niż wartości udostępnione przez zmienną makra MQMD_DEFAULT. (Nie zawsze jest to konieczne; w niektórych środowiskach można użyć wartości w postaci łańcucha w obu sytuacjach. Można jednak użyć formularza tablicy dla deklaracji, ponieważ jest to wymagane w celu zapewnienia zgodności z językiem programowania C + +.)

Wartości początkowe dla struktur dynamicznych

Jeśli wymagana jest zmienna liczba instancji struktury, instancje są zwykle tworzone w pamięci głównej uzyskanej dynamicznie za pomocą funkcji `calloc` lub `malloc` . Aby zainicjować pola w takich strukturach, należy wziąć pod uwagę następującą technikę:

1. Zadeklaruj instancję struktury za pomocą odpowiedniej zmiennej makra MQxxx_DEFAULT w celu zainicjowania struktury. Ta instancja staje się modelem dla innych instancji:

   ```
   MQMD Model = {MQMD_DEFAULT}; /* declare model instance */
   ```

   Słowa kluczowe `static` lub `auto` mogą być zakodowane w deklaracji w celu nadania instancji modelu statycznego lub dynamicznego czasu życia (zgodnie z wymaganiami).
2. Użyj funkcji `calloc` lub `malloc` , aby uzyskać pamięć dla dynamicznej instancji struktury:

   ```
   PMQMD Instance;
   Instance = malloc(sizeof(MQMD)); /* get storage for dynamic instance */
   ```
3. Użyj funkcji `memcpy` , aby skopiować instancję modelu do instancji dynamicznej:

   ```
   memcpy(Instance,&Model,sizeof(MQMD)); /* initialize dynamic instance */
   ```

Użyj z C++

W przypadku języka programowania C++ pliki nagłówkowe zawierają następujące dodatkowe instrukcje, które są dołączane tylko wtedy, gdy używany jest kompilator C + +:

```
#ifdef __cplusplus
  extern "C" {
```

```
#endif

/* rest of header file */

#ifdef __cplusplus
  }
#endif
```

Konwencje notacji

Te informacje przedstawiają sposób wywoływania funkcji i deklarowania parametrów.

W niektórych przypadkach parametry są tablicami o nieustalonej wielkości. W przypadku tych wartości do reprezentowania stałej liczbowej używana jest mała litera n. Podczas kodowania deklaracji dla tego parametru należy zastąpić wartość n wymaganą wartością liczbową.

Programowanie w języku COBOL

Informacje pomocne w korzystaniu z interfejsu MQI z języka programowania COBOL.

- "Pliki COPY" na stronie 268
- "Struktury" na stronie 269
- "Wskaźniki" na stronie 270
- "Stałe nazwane" na stronie 270
- "Konwencje notacji" na stronie 271

Pliki COPY

Dostępne są różne pliki COPY, które ułatwiają pisanie aplikacji w języku COBOL korzystających z interfejsu MQI. Istnieją dwa pliki zawierające stałe nazwane i dwa pliki dla każdej struktury.

Każda struktura jest udostępniana w dwóch formach: formie z wartościami początkowymi i formie bez:

- Użyj struktur z wartościami początkowymi w sekcji WORKING-STORAGE SECTION programu COBOL; są one zawarte w plikach COPY z nazwami z przyrostkiem V (dla wartości).
- Struktur bez wartości początkowych należy używać w sekcji `LINKAGE SECTION` programu w języku COBOL. Są one zawarte w plikach COPY z nazwami z przyrostkiem L (dla Linkage).

Poniższa tabela zawiera podsumowanie plików COPY. Nie wszystkie wymienione pliki są dostępne we wszystkich środowiskach.

Tabela 465. Pliki COPY języka COBOL

Plik (z wartościami początkowymi)	**Plik (bez wartości początkowych)**	**Spis treści**
CMQAIRV	KMQAIRL	Rekord informacji uwierzytelniającej
CMQBOV	CMQBOL	Struktura opcji początku
CMQCIHV	CMQCIHL	Struktura nagłówka informacji CICS
CMQCNOV	CMQCNOL	Struktura opcji połączenia
CMQDHV	CMQDHL	Struktura nagłówka dystrybucji
CMQDLHV	CMQDLHL	Struktura nagłówka niedostarczonego komunikatu
CMQDXPV	CMQDXPL	Struktura parametru wyjścia konwersji danych
CMQGMOV	CMQGMOL	Pobierz strukturę opcji komunikatu
CMQIIHV	CMQIIHL	Struktura nagłówka informacji IMS

Tabela 465. Pliki COPY języka COBOL (kontynuacja)		
Plik (z wartościami początkowymi)	**Plik (bez wartości początkowych)**	**Spis treści**
CMQMDV	CMQMDLLanguage	Struktura deskryptora komunikatu
CMQMDEV	CMQMDEL	Struktura rozszerzenia deskryptora komunikatu
CMQMD1V	CMQMD1L	Struktura deskryptora komunikatu, wersja 1
CMQODV	CMQODL	Struktura deskryptora obiektu
KMQORV	KMQORL	Struktura rekordu obiektu
CMQPMOV	CMQPMOL	Struktura opcji umieszczania komunikatu
CMQRFHV	CMQRFHL	Reguły i struktura nagłówka formatowania
CMQRFH2V	CMQRFH2L	Reguły i formatowanie struktury nagłówka w wersji 2
CMQRMHV	CMQRMHL	Struktura nagłówka komunikatu odwołania
CMQRRV	KMQRRL	Struktura rekordu odpowiedzi
CMQSCOV,	Komenda CMQSCOL	Opcje konfiguracyjne TLS
CMQTMV	CMQTML	Struktura komunikatu wyzwalacza
CMQTMCV	CMQTMCL,	Struktura komunikatu wyzwalacza (format znakowy)
CMQTMC2V	CMQTMC2L	Struktura komunikatu wyzwalacza (format znakowy) wersja 2
CMQWIHV	CMQWIHL	Struktura nagłówka informacji o pracy
CMQXQHV	CMQXQHL	Struktura nagłówka kolejki transmisji
CMQV	-	Stałe nazwane dla głównego interfejsu MQI
CMQXV,	-	Stałe nazwane dla wyjścia konwersji danych
CMQMD2V	CMQMD2L	Struktura deskryptora komunikatu, wersja 2

Struktury

W pliku COPY każda deklaracja struktury rozpoczyna się od elementu level-10 . Umożliwia to zadeklarowanie kilku instancji struktury przez zakodowanie deklaracji level-01 , a następnie użycie instrukcji COPY w celu skopiowania pozostałej części deklaracji struktury. Aby odwołać się do odpowiedniej instancji, należy użyć parametru 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.
```

Wyrównaj struktury na odpowiednich granicach. W przypadku użycia instrukcji COPY w celu uwzględnienia struktury po elemencie, który nie jest elementem level-01 , upewnij się, że struktura rozpoczyna się od odpowiedniego przesunięcia od początku elementu level-01 . Większość struktur MQI wymaga wyrównania 4-bajtowego. Wyjątki od tego wyrównania to MQCNO, MQOD i MQPMO, które wymagają wyrównania 16-bajtowego w systemie IBM i.

W tej sekcji nazwy pól w strukturach są wyświetlane bez przedrostka. W języku COBOL: nazwy pól są poprzedzone nazwą struktury, po której następuje łącznik. Jeśli jednak nazwa struktury kończy się cyfrą, co oznacza, że struktura jest drugą lub późniejszą wersją oryginalnej struktury, cyfra liczbowa jest pomijana w przedrostku. Nazwy pól w języku COBOL są wyświetlane wielkimi literami (w razie potrzeby można używać małych lub mieszanych liter). Na przykład pole *MsgType* opisane w sekcji “MQMD-deskryptor komunikatu” na stronie 432 staje się w języku COBOL polem MQMD-MSGTYPE.

Struktury przyrostka V są deklarowane z wartościami początkowymi dla wszystkich pól; należy ustawić tylko te pola, w których wymagana jest wartość inna niż podana wartość początkowa.

Wskaźniki

Niektóre struktury muszą adresować opcjonalne dane, które mogą być nieciągłe w strukturze, takie jak rekordy MQOR i MQRR adresowane przez strukturę MQOD.

Aby uwzględnić te dane opcjonalne, struktury zawierają pola zadeklarowane z typem danych wskaźnika. Jednak język COBOL nie obsługuje typu danych wskaźnika we wszystkich środowiskach. Z tego powodu dane opcjonalne mogą być również adresowane przy użyciu pól, które zawierają przesunięcie danych od początku struktury.

Jeśli aplikacja ma być używana między środowiskami, należy sprawdzić, czy typ danych wskaźnika jest dostępny we wszystkich planowanych środowiskach. Jeśli nie, aplikacja musi adresować opcjonalne dane przy użyciu pól przesunięcia zamiast pól wskaźnika.

W środowiskach, w których wskaźniki nie są obsługiwane, należy zadeklarować pola wskaźnika jako łańcuchy bajtowe o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null. Nie należy zmieniać tej wartości początkowej, jeśli używane są pola przesunięcia.

Stałe nazwane

W tej informacji wyświetlane są nazwy stałych zawierające znak podkreślenia (_) jako część nazwy. W języku COBOL zamiast znaku podkreślenia należy użyć znaku łącznika (-).

Stałe, które mają wartości łańcucha znaków, używają pojedynczego znaku cudzysłowu jako separatora łańcucha ('). W niektórych środowiskach może być konieczne określenie odpowiedniej opcji kompilatora, aby kompilator zaakceptował pojedynczy cudzysłów jako ogranicznik łańcucha zamiast podwójnego cudzysłowu.

Stałe nazwane są deklarowane w plikach COPY jako elementy level-10 . Aby użyć stałych, należy jawnie zadeklarować element level-01 , a następnie użyć instrukcji COPY w celu skopiowania deklaracji stałych:

```
* Declare a structure to hold the constants
 01  MY-MQ-CONSTANTS.
     COPY CMQV.
```

Powyższa metoda powoduje, że stałe zajmują pamięć w programie, nawet jeśli nie są przywoływane. W przypadku uwzględnienia stałych w wielu oddzielnych programach w tej samej jednostce uruchamiania, istnieje wiele kopii stałych, które niepotrzebnie zużywają pamięć główną. Aby uniknąć tego efektu, należy użyć jednej z następujących technik:

- Dodaj klauzulę GLOBAL do deklaracji level-01 :

  ```
  * Declare a global structure to hold the constants
   01  MY-MQ-CONSTANTS GLOBAL.
       COPY CMQV.
  ```

  Spowoduje to przydzielenie pamięci tylko dla jednego zestawu stałych w jednostce uruchamiania. Stałe mogą być jednak przywoływane przez dowolny program w jednostce uruchamiania, a nie tylko przez program, który zawiera deklarację level-01 .

  Uwaga: Klauzula GLOBAL nie jest obsługiwana we wszystkich środowiskach.

- Ręcznie kopiuj do każdego programu tylko te stałe, do których odwołuje się ten program. Nie należy używać instrukcji COPY do kopiowania wszystkich stałych do programu.

Konwencje notacji

W poprzednich sekcjach tego tematu przedstawiono sposób wywoływania wywołań i deklarowania parametrów. W niektórych przypadkach parametry są tabelami lub łańcuchami znaków, których wielkość nie jest ustalona. W przypadku tych wartości do reprezentowania stałej liczbowej używana jest mała litera n. Podczas kodowania deklaracji dla tego parametru należy zastąpić wartość n wymaganą wartością liczbową.

Programowanie High Level Assembler

Informacje ułatwiające użycie interfejsu MQI z języka programowania System/390 Assembler.

- "Makra" na stronie 271
- "Struktury" na stronie 272
- "Makro CMQVERA" na stronie 272
- "Konwencje notacji" na stronie 272

Makra

Istnieją dwa makra dla stałych nazwanych i jedno makro dla każdej struktury. Te pliki zostały podsumowane w poniższej tabeli.

Tabela 466. Makra asemblera

Plik	**Spis treści**
KMQA	Stałe nazwane (równoważne) dla głównego interfejsu MQI
CMQCIHA	Struktura nagłówka informacji CICS
CMQCNOA	Struktura opcji połączenia
CMQDLHA	Struktura nagłówka niedostarczonego komunikatu
CMQDXPA	Struktura parametru wyjścia konwersji danych
CMQGMOA	Pobierz strukturę opcji komunikatu
CMQIIHA	Struktura nagłówka informacji IMS
CMQMDA	Struktura deskryptora komunikatu
CMQMDEA	Struktura rozszerzenia deskryptora komunikatu
CMQODA	Struktura deskryptora obiektu
CMQPMOA	Struktura opcji umieszczania komunikatu
CMQRFHA	Reguły i struktura nagłówka formatowania
CMQRFH2A	Reguły i formatowanie struktury nagłówka w wersji 2
CMQRMHA	Struktura nagłówka komunikatu odwołania
CMQTMA	Struktura komunikatu wyzwalacza
CMQTMC2A	Struktura komunikatu wyzwalacza (format znakowy) wersja 2
KMQVERA	Kontrola wersji struktury
CMQWIHA	Struktura nagłówka informacji o pracy
Usługa CMQXA	Stałe nazwane dla wyjścia konwersji danych

Tabela 466. Makra asemblera (kontynuacja)	
Plik	**Spis treści**
KMQXPA	Struktura parametru wyjścia przecięcia funkcji API
CMQXQHA	Struktura nagłówka kolejki transmisji

Struktury

Struktury są generowane przez makra, które mają różne parametry sterujące działaniem makra. Patrz: “Struktury” na stronie 272

Makro CMQVERA

To makro umożliwia ustawienie wartości domyślnej, która ma być używana dla parametru DCLVER w makrach struktury.

Wartość określona przez CMQVERA jest używana przez makro struktury tylko wtedy, gdy podczas wywoływania makra struktury pominięto parametr DCLVER . Wartość domyślną ustawia się, kodując makro CMQVERA za pomocą parametru DCLVER :

DCLVER=CURRENT
Wersja domyślna jest ustawiona na bieżącą (najnowszą) wersję.

DCLVER=OKREŚLONY
Wersja domyślna jest ustawiana na wersję określoną przez parametr VERSION .

Należy podać parametr **DCLVER** , a wartość musi być zapisana wielkimi literami. Wartość ustawiona przez CMQVERA pozostaje wartością domyślną do następnego wywołania CMQVERA lub do końca zespołu. W przypadku pominięcia CMQVERA wartością domyślną jest DCLVER=CURRENT.

Konwencje notacji

W innych tematach opisano sposób wywoływania wywołań i deklarowania parametrów. W niektórych przypadkach parametry są tablicami lub łańcuchami znaków o nieustalonym rozmiarze, dla którego do reprezentowania stałej liczbowej używana jest mała litera n. Podczas kodowania deklaracji dla tego parametru należy zastąpić wartość n wymaganą wartością liczbową.

Struktury
Struktury są generowane przez makra, które mają różne parametry sterujące działaniem makra.

Uwaga: Od czasu do czasu wprowadzane są nowe wersje struktur IBM MQ . Dodatkowe pola w nowej wersji mogą spowodować, że struktura, która wcześniej była mniejsza niż 256 bajtów, stanie się większa niż 256 bajtów. Z tego powodu należy napisać instrukcje asemblera, które mają na celu skopiowanie struktury IBM MQ lub ustawienie struktury IBM MQ na wartość null, aby działać poprawnie ze strukturami, które mogą być większe niż 256 bajtów. Alternatywnie można użyć parametru makra DCLVER lub makra CMQVERA z parametrem VERSION w celu zadeklarowania konkretnej wersji struktury.

- “Określanie nazwy struktury” na stronie 272
- “Określanie formy struktury” na stronie 273
- “Sterowanie wersją struktury” na stronie 273
- “Deklarowanie jednej struktury osadzonej w innej” na stronie 273
- “Określanie wartości początkowych dla zmiennych” na stronie 274
- “Sterowanie listingiem” na stronie 274

Określanie nazwy struktury

Aby zadeklarować więcej niż jedną instancję struktury, makro poprzedza nazwę każdego pola w strukturze łańcuchem, który można określić przez użytkownika, i znakiem podkreślenia.

Użyty łańcuch jest etykietą określoną podczas wywoływania makra. Jeśli nie określono etykiety, do utworzenia przedrostka używana jest nazwa struktury:

```
* Declare two object descriptors
         CMQODA  ,        Prefix used="MQOD_" (the default)
MY_MQOD  CMQODA  ,        Prefix used="MY_MQOD_"
```

Deklaracje struktury przedstawione w tej sekcji korzystają z domyślnego przedrostka.

Określanie formy struktury

Deklaracje struktury mogą być generowane przez makro w jednej z dwóch form sterowanych przez parametr DSECT :

DSECT = TAK
Instrukcja DSECT asemblera jest używana do uruchamiania nowej sekcji danych; definicja struktury następuje bezpośrednio po instrukcji DSECT . Etykieta w wywołaniu makra jest używana jako nazwa sekcji danych; jeśli nie określono etykiety, używana jest nazwa struktury.

DSECT = NIE
Instrukcje DC asemblera są używane do definiowania struktury w bieżącej pozycji w procedurze. Pola są inicjowane wartościami, które można określić, kodując odpowiednie parametry w wywołaniu makra. Pola, dla których nie określono wartości w wywołaniu makra, są inicjowane wartościami domyślnymi.

Podana wartość musi być zapisana wielkimi literami. Jeśli parametr DSECT nie zostanie podany, przyjmowany jest parametr DSECT = NO .

Sterowanie wersją struktury

Domyślnie makra zawsze deklarują najnowszą wersję każdej struktury.

Chociaż można użyć parametru makra VERSION do określenia wartości pola *Version* w strukturze, parametr ten definiuje początkową wartość pola *Version* i nie steruje wersją struktury rzeczywiście zadeklarowanej. Aby sterować wersją zadeklarowanej struktury, należy użyć parametru DCLVER :

DCLVER=CURRENT
Zadeklarowana wersja jest bieżącą (najnowszą) wersją.

DCLVER=OKREŚLONY
Zadeklarowana wersja jest wersją określoną przez parametr VERSION . Jeśli parametr VERSION zostanie pominięty, wartością domyślną jest wersja 1.

Jeśli zostanie podany parametr VERSION , wartość musi być samodefiniującą się stałą numeryczną lub stałą nazwaną dla wymaganej wersji (na przykład MQCNO_VERSION_3). Jeśli zostanie podana inna wartość, struktura zostanie zadeklarowana jako DCLVER=CURRENT , nawet jeśli wartość zmiennej VERSION jest tłumaczona na poprawną wartość.

Podana wartość musi być zapisana wielkimi literami. Jeśli parametr DCLVER zostanie pominięty, użyta wartość zostanie pobrana z globalnej zmiennej makra MQDCLVER . Tę zmienną można ustawić przy użyciu makra CMQVERA.

Deklarowanie jednej struktury osadzonej w innej

Aby zadeklarować jedną strukturę jako komponent innej struktury, należy użyć parametru NESTED :

NESTED=TAK
Deklaracja struktury jest zagnieżdżona w innej.

NESTED=NIE
Deklaracja struktury nie jest zagnieżdżona w innej.

Podana wartość musi być zapisana wielkimi literami. Jeśli parametr NESTED zostanie pominięty, zostanie przyjęta wartość NESTED=NO .

Określanie wartości początkowych dla zmiennych

Określ wartość, która ma być używana do inicjowania pola w strukturze, kodując nazwę tego pola (bez przedrostka) jako parametr w wywołaniu makra, wraz z wymaganą wartością.

Na przykład, aby zadeklarować strukturę deskryptora komunikatu z polem *MsgType* zainicjowanym za pomocą MQMT_REQUEST i polem *ReplyToQ* zainicjowanym za pomocą łańcucha "MY_REPLY_TO_QUEUE", należy użyć następującej składni:

```
MY_MQMD  CMQMDA  MSGTYPE=MQMT_REQUEST,                              X
               REPLYTOQ=MY_REPLY_TO_QUEUE
```

W przypadku określenia stałej nazwanej (equate) jako wartości w wywołaniu makra należy użyć makra CMQA w celu zdefiniowania stałej nazwanej. Wartości łańcuchów znakowych nie należy ujmować w apostrofy.

Sterowanie listingiem

Sterowanie wyglądem deklaracji struktury w listingu asemblera za pomocą parametru LIST :

LIST = TAK
Deklaracja struktury zostanie wyświetlona na liście asemblera.

LISTA = NIE
Deklaracja struktury nie pojawia się na listingu asemblera.

Podana wartość musi być zapisana wielkimi literami. Jeśli parametr LIST zostanie pominięty, przyjmowany jest parametr LIST = NO .

MQAIR-rekord informacji uwierzytelniającej

Struktura MQAIR umożliwia aplikacji działającej jako IBM MQ MQI client określenie informacji o elemencie uwierzytelniającym, który ma być używany dla połączenia klienta. Struktura jest parametrem wejściowym wywołania MQCONNX.

Dostępność

Struktura MQAIR jest dostępna dla następujących klientów:

- AIX
- Linux
- Windows

Zestaw znaków i kodowanie

Dane w usłudze MQAIR muszą być w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one udostępniane przez atrybut menedżera kolejek systemu **CodedCharSetId** i parametr MQENC_NATIVE.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 467. Pola w MQAIR

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	Identyfikator struktury MQAIR_STRUC_ID	'AIR¬'

Tabela 467. Pola w MQAIR (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
Wersja (numer wersji struktury)	MQAIR_VERSION_1	1
AuthInfoTyp (typ informacji uwierzytelniających)	MQAIT_CRL_LDAP	1
AuthInfoConnName (nazwa połączenia z serwerem CRL LDAP)	Brak	Pusty łańcuch lub odstępy
LDAPUserNamePtr (adres nazwy użytkownika LDAP)	Brak	Pusty wskaźnik lub puste bajty
LDAPUserNamePrzesunięcie (przesunięcie nazwy użytkownika LDAP od początku MQSCO)	Brak	0
LDAPUserNameDługość (długość nazwy użytkownika LDAP)	Brak	0
LDAPPassword (hasło dostępu do serwera LDAP)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość w polu *Wersja* jest mniejsza niż MQAIR_VERSION_2.		
OCSPResponderURL (URL , pod którym można skontaktować się z responderem OCSP)	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraParametr MQAIR_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQAIR MyAIR = {MQAIR_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQAIR

```
typedef struct tagMQAIR MQAIR;
struct tagMQAIR {
  MQCHAR4    StrucId;             /* Structure identifier */
  MQLONG     Version;             /* Structure version number */
  MQLONG     AuthInfoType;        /* Type of authentication
                                     information */
  MQCHAR264  AuthInfoConnName;    /* Connection name of CRL LDAP
                                     server */
  PMQCHAR    LDAPUserNamePtr;     /* Address of LDAP user name */
  MQLONG     LDAPUserNameOffset;  /* Offset of LDAP user name from start
                                     of MQAIR structure */
  MQLONG     LDAPUserNameLength;  /* Length of LDAP user name */
  MQCHAR32   LDAPPassword;        /* Password to access LDAP server */
  MQCHAR256  OCSPResponderURL;    /* URL of OCSP responder */

};
```

Deklaracja języka COBOL dla MQAIR

```
**   MQAIR structure
  10 MQAIR.
**    Structure identifier
```

```
   15 MQAIR-STRUCID            PIC X(4).
**    Structure version number
   15 MQAIR-VERSION            PIC S9(9) BINARY.
**    Type of authentication information
   15 MQAIR-AUTHINFOTYPE       PIC S9(9) BINARY.
**    Connection name of CRL LDAP server
   15 MQAIR-AUTHINFOCONNNAME   PIC X(264).
**    Address of LDAP user name
   15 MQAIR-LDAPUSERNAMEPTR    POINTER.
**    Offset of LDAP user name from start of MQAIR structure
   15 MQAIR-LDAPUSERNAMEOFFSET PIC S9(9) BINARY.
**    Length of LDAP user name
   15 MQAIR-LDAPUSERNAMELENGTH PIC S9(9) BINARY.
**    Password to access LDAP server
   15 MQAIR-LDAPPASSWORD       PIC X(32).
**    URL of OCSP responder
   15 MQAIR-OCSPRESPONDERURL   PIC X(256).
```

Deklaracja Visual Basic dla MQAIR

```
Type MQAIR
  StrucId            As String*4   'Structure identifier'
  Version            As Long       'Structure version number'
  AuthInfoType       As Long       'Type of authentication information'
  AuthInfoConnName   As String*264 'Connection name of CRL LDAP server'
  LDAPUserNamePtr    As MQPTR      'Address of LDAP user name'
  LDAPUserNameOffset As Long       'Offset of LDAP user name from start'
                                   'of MQAIR structure'
  LDAPUserNameLength As Long       'Length of LDAP user name'
  LDAPPassword       As String*32  'Password to access LDAP server'
End Type
```

StrucId (MQCHAR4) dla MQAIR

Jest to identyfikator struktury struktury rekordu informacji uwierzytelniającej. Jest to zawsze pole wejściowe. Jego wartością jest MQAIR_STRUC_ID.

Wartość musi być następująca:

Identyfikator struktury MQAIR_STRUC_ID
Identyfikator rekordu informacji uwierzytelniającej.

Dla języka programowania C zdefiniowana jest również stała MQAIR_STRUC_ID_ARRAY. Ta wartość jest taka sama jak wartość MQAIR_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQAIR

Jest to numer wersji struktury rekordu informacji uwierzytelniającej. Jest to zawsze pole wejściowe.

Wartość musi być jedną z następujących wartości:

MQAIR_VERSION_1
Version-1 rekord informacji uwierzytelniającej.

Jest to wartość początkowa tego pola.

MQAIR_VERSION_2
Version-2 rekord informacji uwierzytelniającej.

Następująca stała określa numer wersji bieżącej:

MQAIR_CURRENT_VERSION
Bieżąca wersja rekordu informacji uwierzytelniającej.

AuthInfoTyp (MQLONG) dla MQAIR

Jest to typ informacji uwierzytelniających zawartych w rekordzie.

Wartością może być jeden z dwóch następujących parametrów:

MQAIT_CRL_LDAP
Sprawdzanie odwołań certyfikatów przy użyciu serwera LDAP.

MQAIT_OCSP
Sprawdzanie odwołań certyfikatów przy użyciu protokołu OCSP.

Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_TYPE_ERROR.

Jest to pole wejściowe. Wartością początkową tego pola jest MQAIT_CRL_LDAP.

AuthInfoConnName (MQCHAR264) for MQAIR

Jest to nazwa hosta lub adres sieciowy hosta, na którym działa serwer LDAP. Po tym numerze można podać opcjonalny numer portu ujęty w nawiasy. Domyślny numer portu to 389.

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_CONN_NAME_ERROR.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_AUTH_INFO_CONN_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C oraz puste znaki w innych językach programowania.

LDAPUserNamePtr (PMQCHAR) dla MQAIR

Jest to nazwa użytkownika LDAP.

Składa się ona z nazwy wyróżniającej użytkownika, który próbuje uzyskać dostęp do serwera CRL LDAP. Jeśli wartość jest krótsza niż długość określona przez parametr *LDAPUserNameLength*, należy ją zakończyć znakiem o kodzie zero lub dopełnić spacjami do długości *LDAPUserNameLength*. Pole jest ignorowane, jeśli parametr *LDAPUserNameLength* ma wartość zero.

Nazwę użytkownika LDAP można podać na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *LDAPUserNamePtr*

  W takim przypadku aplikacja może zadeklarować łańcuch, który jest oddzielony od struktury MQAIR, i ustawić parametr *LDAPUserNamePtr* na adres łańcucha.

  Należy rozważyć użycie języka *LDAPUserNamePtr* dla języków programowania obsługujących typ danych wskaźnika w sposób, który jest przenośny w różnych środowiskach (na przykład w języku programowania C).
- Używając pola przesunięcia *LDAPUserNameOffset*

  W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą strukturę MQSCO, po której następuje tablica rekordów MQAIR, po której następuje łańcuch nazwy użytkownika LDAP, i ustawić wartość *LDAPUserNameOffset* na przesunięcie odpowiedniego łańcucha nazwy od początku struktury MQAIR. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być ustawiona w MQLONG (najbardziej restrykcyjnym językiem programowania jest COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

  Należy rozważyć użycie języka *LDAPUserNameOffset* dla języków programowania, które nie obsługują typu danych wskaźnika lub implementują typ danych wskaźnika w sposób, który może nie być przenośny w różnych środowiskach (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki należy używać tylko jednej z następujących metod: *LDAPUserNamePtr* i *LDAPUserNameOffset* . Wywołanie nie powiodło się z kodem przyczyny MQRC_LDAP_USER_NAME_ERROR, jeśli obie wartości są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

LDAPUserNamePrzesunięcie (MQLONG) dla MQAIR

Jest to przesunięcie w bajtach nazwy użytkownika LDAP od początku struktury MQAIR.

Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli parametr *LDAPUserNameLength* ma wartość zero.

Do określenia nazwy użytkownika LDAP można użyć wartości *LDAPUserNamePtr* lub *LDAPUserNameOffset* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola *LDAPUserNamePtr* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

LDAPUserNameDługość (MQLONG) dla MQAIR

Jest to długość w bajtach nazwy użytkownika LDAP adresowanej przez pole *LDAPUserNamePtr* lub *LDAPUserNameOffset* .

Wartość musi być z zakresu od 0 do MQ_DISTINGUISHED_NAME_LENGTH. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_LDAP_USER_NAME_LENGTH_ERR.

Jeśli używany serwer LDAP nie wymaga nazwy użytkownika, należy ustawić w tym polu wartość zero.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

LDAPPassword (MQCHAR32) dla MQAIR

Jest to hasło wymagane do uzyskania dostępu do serwera CRL LDAP. Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola.

Jeśli serwer LDAP nie wymaga hasła lub jeśli pominięto nazwę użytkownika LDAP, parametr *LDAPPassword* musi mieć wartość null lub być pusty. Jeśli nazwa użytkownika LDAP zostanie pominięta, a parametr *LDAPPassword* nie ma wartości NULL ani nie jest pusty, wywołanie nie powiedzie się z kodem przyczyny MQRC_LDAP_PASSWORD_ERROR.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_LDAP_PASSWORD_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C oraz puste znaki w innych językach programowania.

OCSPResponderURL (MQCHAR256) dla MQAIR

W przypadku struktury MQAIR reprezentującej szczegóły połączenia dla programu odpowiadającego OCSP pole to zawiera URL , pod którym można nawiązać kontakt z programem odpowiadającym.

Wartością tego pola jest HTTP URL. To pole ma pierwszeństwo przed URL w rozszerzeniu certyfikatu AuthorityInfoAccess (AIA).

Wartość jest ignorowana, chyba że spełnione są oba poniższe warunki:

- Struktura MQAIR jest w wersji 2 lub nowszej (pole Wersja ma wartość MQAIR_VERSION_2 lub większą).
- Pole typu AuthInfojest ustawione na wartość MQAIT_OCSP.

Jeśli pole nie zawiera adresu HTTP URL w poprawnym formacie (i nie jest ignorowane), wywołanie MQCONNX kończy się niepowodzeniem z kodem przyczyny MQRC_OCSP_URL_ERROR.

W tym polu jest rozróżniana wielkość liter. Musi zaczynać się od łańcucha http:// pisanego małymi literami. W pozostałych URL może być rozróżniana wielkość liter, w zależności od implementacji serwera OCSP.

To pole nie podlega konwersji danych.

MQBMHO-Opcje obsługi bufor-komunikat

Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia uchwytów komunikatów z buforów. Struktura jest parametrem wejściowym wywołania MQBUFMH.

Zestaw znaków i kodowanie

Dane w obiekcie MQBMHO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 468. Pola w MQBMHO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQBMHO_STRUC_ID	'BMHO'
Wersja (numer wersji struktury)	MQBMHO_VERSION_1	1
Opcje (opcje sterujące działaniem komendy MQBMHO)	MQBMHO_NONE	0

Uwagi:

1. W języku programowania C: zmienna makraMQBMHO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQBMHO MyBMHO = {MQBMHO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQBMHO

```
typedef struct tagMQBMHO MQBMHO;
struct tagMQBMHO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQBUFMH */
};
```

Deklaracja języka COBOL dla MQBMHO

```
**   MQBMHO structure
  10 MQBMHO.
**    Structure identifier
   15 MQBMHO-STRUCID               PIC X(4).
**    Structure version number
   15 MQBMHO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQBUFMH
   15 MQBMHO-OPTIONS               PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQBMHO

```
Dcl
 1 MQBMHO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action
                                      of MQBUFMH */
```

Deklaracja High Level Assembler dla MQBMHO

```
MQBMHO                   DSECT
MQBMHO_STRUCID           DS   CL4  Structure identifier
MQBMHO_VERSION           DS   F    Structure version number
MQBMHO_OPTIONS           DS   F    Options that control the
*                                   action of MQBUFMH
```

```
MQBMHO_LENGTH               EQU  *-MQBMHO
MQBMHO_AREA                 DS   CL(MQBMHO_LENGTH)
```

StrucId (MQCHAR4) dla MQBMHO dla MQBMHO

Jest to identyfikator struktury struktury buforu do obsługi komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQBMHO_STRUC_ID.

Wartość musi być następująca:

MQBMHO_STRUC_ID
Identyfikator struktury uchwytu buforu do komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQBMHO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQBMHO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQBMHO dla MQBMHO

Jest to numer wersji struktury obsługi buforu do komunikatu. Jest to zawsze pole wejściowe.

Wartość musi być następująca:

MQBMHO_VERSION_1
Numer wersji struktury uchwytu buforu do komunikatu.

Następująca stała określa numer wersji bieżącej:

MQBMHO_CURRENT_VERSION
Bieżąca wersja struktury uchwytu buforu do komunikatu.

Opcje (MQLONG) dla MQBMHO

Struktura uchwytu buforu do komunikatu-pole Opcje

Możliwe wartości:

MQBMHO_DELETE_PROPERTIES
Właściwości, które są dodawane do uchwytu komunikatu, są usuwane z buforu. Jeśli wywołanie nie powiedzie się, nie zostaną usunięte żadne właściwości.

Opcje domyślne: Jeśli opisywana opcja nie jest potrzebna, należy użyć następującej opcji:

MQBMHO_NONE
Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQBMHO_DELETE_PROPERTIES.

V 9.3.0 MQBNO-opcje równoważenia

Poniższa tabela zawiera podsumowanie pól w strukturze.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 469. Pola w MQBNO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQBNO_STRUC_ID	'BNO¬'
Wersja (numer wersji struktury)	MQBNO_VERSION_1	1
ApplicationType (typ opcji równoważenia ustawiony w strukturze)	MQBNO_VALTYPE_SIMPLE	0

Tabela 469. Pola w MQBNO (kontynuacja)

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
Timeout (limit czasu, po którym ponowne równoważenie może przerwać działanie aplikacji)	MQBNO_TIMEOUT_AS_DEFAULT	0
BalanceOptions (opcje równoważenia ustawione przez aplikację wydającą)	MQBNO_OPTIONS_NONE	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C zmienna makra MQBNO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQBNO MyBNO = {MQBNO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQBNO

```
typedef struct tagMQBNO MQBNO;
struct tagMQBNO {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQLONG     Type;              /* Type of balancing options set in the
                                   structure */
  MQLONG     Timeout;           /* Timeout after which re-balancing might
                                   interrupt application activity */
  MQLONG     BalanceOptions;    /* Balancing options set by the issuing
                                   application */
};
```

Deklaracja języka COBOL dla MQBNO

```
**   MQBNO structure
  10 MQBNO.
**    Structure identifier
   15 MQBNO-STRUCID          PIC X(4).
**    Structure version number
   15 MQBNO-VERSION          PIC S9(9) BINARY.
**    Type of balancing options set in the structure
   15 MQBNO-TYPE             PIC S9(9) BINARY.
**    Timeout after which re-balancing might interrupt application activity
   15 MQBNO-TIMEOUT          PIC S9(9) BINARY.
**    Balancing options set by the issuing application
   15 MQBNO-BALANCEOPTIONS          PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQBNO

```
dcl
 1 MQBNO based,
  3 StrucId          char(4),       /* Structure identifier */
  3 Version          fixed bin(31), /* Structure version number */
  3 Type             fixed bin(31), /* Type of balancing options set in the
                                       structure*/
  3 Timeout          fixed bin(31), /* Timeout after which re-balancing might
                                       interrupt application activity */
  3 BalanceOptions   fixed bin(31), /* Balancing options set by the issuing
                                       application*/
```

Odsyłacze pokrewne

“MQCNO-opcje połączenia” na stronie 321
Struktura MQCNO umożliwia aplikacji określenie opcji dotyczących połączenia z menedżerem kolejek. Struktura jest parametrem wejścia/wyjścia wywołania MQCONNX.

V 9.3.0 *StrucId (MQCHAR4) dla MQBNO*

Jest to identyfikator struktury struktury opcji równoważenia. Jest to zawsze pole wejściowe. Jego wartością początkową jest BNO.

Wartość musi być następująca:

BNO
Identyfikator struktury opcji równoważenia.

Dla języka programowania C zdefiniowana jest również stała MQBNO_STRUC_ID_ARRAY. Ta stała ma taką samą wartość jak BNO, ale jest tablicą znaków zamiast łańcucha.

Należy podać poprawną wartość parametru **StrucId** lub zwrócić wartość MQRC_BNO_ERROR.

V 9.3.0 *Wersja (MQLONG) dla MQBNO*

Jest to numer wersji struktury opcji równoważenia. Jest to zawsze pole wejściowe.

Wartość musi być następująca:

MQBNO_VERSION_1
Numer wersji struktury opcji równoważenia.

Należy podać poprawną wartość parametru **Version** lub zwrócić wartość MQRC_BNO_ERROR.

V 9.3.0 *ApplicationType (MQLONG) dla MQBNO*

Typ opcji równoważenia ustawiony w strukturze.

Możliwe wartości:

MQBNO_BALTYPE_SIMPLE (proste)
Proste równoważenie; oprócz reguł opisanych w sekcji Wpływanie na ponowne równoważenie aplikacji w jednolitych klastrachnie są stosowane żadne szczegółowe reguły.

MQBNO_BALTYPE_REQREP
Równoważenie żądanie-odpowiedź. Po każdym wywołaniu MQPUT oczekiwane jest zgodne wywołanie MQGET dla komunikatu odpowiedzi. Równoważenie jest opóźniane do momentu odebrania takiego komunikatu lub przekroczenia limitu czasu komunikatu żądania.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQBNO_BALTYPE_SIMPLE.

Należy podać tylko jedną wartość w polu **ApplicationType** lub zwracana jest wartość MQRC_BNO_ERROR.

Uwaga: Dodatkowa wartość tego pola opcji MQBNO_BALTYPE_RA_MANAGED jest zarezerwowana dla środowiska IBM MQ Resource Adapter for JEE. Chociaż podanie tej wartości bezpośrednio przez aplikację jest błędem, może ona na przykład zostać zgłoszona podczas wykonywania zapytania o status aplikacji.

V 9.3.0 *Limit czasu (MQLONG) dla MQBNO*

Timeout , po którym ponowne równoważenie może przerwać działanie aplikacji.

Możliwe wartości:

MQBNO_TIMEOUT_AS_DEFAULT
Ustawiona domyślna wartość limitu czasu.

MQBNO_TIMEOUT_IMMEDIATE
Następuje natychmiastowe przekroczenie limitu czasu.

MQBNO_TIMEOUT_NEVER
Nie występuje przekroczenie limitu czasu.

Wartością początkową tego pola jest MQBNO_TIMEOUT_AS_DEFAULT.

Należy podać tylko jedną wartość ze zdefiniowanych wartości lub wartość z zakresu od 0 do 999999999 sekund dla pola **Timeout** lub zwracany jest błąd MQRC_BNO_ERROR.

V 9.3.0 *BalanceOptions (MQLONG) dla MQBNO*

Opcje równoważenia ustawione przez aplikację wydającą.

Możliwe są następujące wartości:

MQBNO_OPTIONS_NONE
Nie ustawiono żadnych opcji

MQBNO_OPTIONS_IGNORE_TRANS
Ustawienie tej opcji umożliwia ponowne równoważenie aplikacji, nawet jeśli są one w trakcie transakcji.

Wartością początkową tego pola jest MQBNO_OPTIONS_NONE.

Można podać dowolną kombinację zdefiniowanych wartości, używając wartości logicznej lub znakowej w polu **BalanceOptions** . Niepoprawne wartości powodują zwrócenie błędu MQRC_BNO_ERROR.

MQBO-opcje początku

Struktura MQBO umożliwia aplikacji określenie opcji związanych z tworzeniem jednostki pracy. Struktura jest parametrem wejścia/wyjścia w wywołaniu komendy MQBEGIN.

Dostępność

Struktura MQBO jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

Struktura MQBO nie jest dostępna dla produktu IBM MQ MQI clients.

Zestaw znaków i kodowanie

Dane w obiekcie MQBO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 470. Pola w obiekcie MQBO dla obiektu MQBO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQBO_STRUC_ID (identyfikator struktury MQBOC)	'BO¬¬'
Wersja (numer wersji struktury)	MQBO_VERSION_1	1

Tabela 470. Pola w obiekcie MQBO dla obiektu MQBO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
Opcje (opcje sterujące działaniem komendy MQBEGIN)	MQBO_BRAK	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQBO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQBO MyBO = {MQBO_DEFAULT};
```

Deklaracje językowe

Deklaracja języka C dla obiektu MQBO

```
typedef struct tagMQBO MQBO;
struct tagMQBO {
  MQCHAR4  StrucId;  /* Structure identifier */
  MQLONG   Version;  /* Structure version number */
  MQLONG   Options;  /* Options that control the action of MQBEGIN */
};
```

Deklaracja języka COBOL dla obiektu MQBO

```
**   MQBO structure
  10 MQBO.
**    Structure identifier
   15 MQBO-STRUCID PIC X(4).
**    Structure version number
   15 MQBO-VERSION PIC S9(9) BINARY.
**    Options that control the action of MQBEGIN
   15 MQBO-OPTIONS PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla obiektu MQBO

```
dcl
 1 MQBO based,
  3 StrucId char(4),       /* Structure identifier */
  3 Version fixed bin(31), /* Structure version number */
  3 Options fixed bin(31); /* Options that control the action of
                              MQBEGIN */
```

Deklaracja Visual Basic dla obiektu MQBO

```
Type MQBO
  StrucId As String*4 'Structure identifier'
  Version As Long     'Structure version number'
  Options As Long     'Options that control the action of MQBEGIN'
End Type
```

StrucId (MQCHAR4) dla obiektu MQBO

Jest to identyfikator struktury struktury opcji początku. Jest to zawsze pole wejściowe. Jego wartością jest MQBO_STRUC_ID.

Wartość musi być następująca:

MQBO_STRUC_ID (identyfikator struktury MQBOC)
Identyfikator struktury opcji początku.

W języku programowania C zdefiniowana jest również stała MQBO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQBO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) obiektu MQBO
Jest to numer wersji struktury opcji początku. Jest to zawsze pole wejściowe.

Wartość musi być następująca:

MQBO_VERSION_1
Numer wersji struktury opcji początku.

Następująca stała określa numer wersji bieżącej:

MQBO_CURRENT_VERSION
Bieżąca wersja struktury opcji początku.

Opcje (MQLONG) obiektu MQBO
To pole jest zawsze polem wejściowym. Jego wartością początkową jest MQBO_NONE.

Wartość musi być następująca:

MQBO_BRAK
Nie określono żadnych opcji.

MQCBC-kontekst wywołania zwrotnego

Struktura MQCBC służy do określania informacji o kontekście przekazywanych do funkcji zwrotnej. Struktura jest parametrem wejścia/wyjścia w wywołaniu procedury konsumenta komunikatów.

Dostępność

Struktura MQCBC jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Wersja

Bieżąca wersja MQCBC to MQCBC_VERSION_2.

Zestaw znaków i kodowanie

Dane w programie MQCBC muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura będzie mieć zestaw znaków i kodowanie klienta.

Pola

Brak wartości początkowych dla struktury **MQCBC** . Struktura jest przekazywana jako parametr do procedury zwrotnej. Menedżer kolejek inicjuje strukturę. Aplikacje nigdy jej nie inicjują.

Uwagi:

- W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.
- Brak wartości początkowych dla struktury MQCBC. Struktura jest przekazywana jako parametr do procedury zwrotnej. Menedżer kolejek inicjuje strukturę. Aplikacje nigdy jej nie inicjują.

Tabela 471. Pola w MQCBC	
Pole	**Opis**
StrucID	Identyfikator struktury
Wersja	Numer wersji struktury
CallType	Dlaczego funkcja została wywołana
Hobby	Uchwyt obiektu
CallbackArea	Pole dla funkcji zwrotnej, która ma być używana
ConnectionArea	Pole dla funkcji zwrotnej, która ma być używana
CompCode	Kod zakończenia
Powód	Kod przyczyny
STATE	Wskazanie stanu obecnego konsumenta
DataLength	Długość komunikatu
BufferLength	Długość buforu komunikatów w bajtach
Flagi	Flagi ogólne
Uwaga: Pozostałe pole jest ignorowane, jeśli wartość w polu Wersja jest mniejsza niż MQCBC_VERSION_2	
ReconnectDelay	Liczba milisekund przed próbą ponownego połączenia

Deklaracje językowe

Deklaracja C dla MQCBC

```
typedef struct tagMQCBC MQCBC;
struct tagMQCBC {
   MQCHAR4   StrucId;                /* Structure identifier */
   MQLONG    Version;                /* Structure version number */
   MQLONG    CallType;               /* Why Function was called */
   MQHOBJ    Hobj;                   /* Object Handle */
   MQPTR     CallbackArea;           /* Callback data passed to the function */
   MQPTR     ConnectionArea;         /* MQCTL data area passed to the function */
   MQLONG    CompCode;               /* Completion Code */
   MQLONG    Reason;                 /* Reason Code */
   MQLONG    State;                  /* Consumer State */
   MQLONG    DataLength;             /* Message Data Length */
   MQLONG    BufferLength;           /* Buffer Length */
   MQLONG    Flags;                  /* Flags containing information about
                                        this consumer */
  /* Ver:1 */
   MQLONG    ReconnectDelay;         /* Number of milliseconds before */
  /* Ver:2 */  };                   /* reconnect attempt */
```

Deklaracja języka COBOL dla MQCBC

```
** MQCBC structure
 10  MQCBC.
** Structure Identifier
```

```
 15  MQCBC-STRUCID                       PIC X(4).
** Structure Version
 15  MQCBC-VERSION                       PIC S9(9) BINARY.
 ** Call Type
 15  MQCBC-CALLTYPE                      PIC S9(9) BINARY.
** Object Handle
 15  MQCBC-HOBJ                          PIC S9(9) BINARY.
** Callback User Area
 15  MQCBC-CALLBACKAREA                  POINTER
** Connection Area
 15  MQCBC-CONNECTIONAREA                POINTER
** Completion Code
 15  MQCBC-COMPCODE                      PIC S9(9) BINARY.
** Reason Code
 15  MQCBC-REASON                        PIC S9(9) BINARY.
** Consumer State
 15  MQCBC-STATE                         PIC S9(9) BINARY.
** Data Length
 15  MQCBC-DATALENGTH                    PIC S9(9) BINARY.
** Buffer Length
 15  MQCBC-BUFFERLENGTH                  PIC S9(9) BINARY.
** Flags
 15  MQCBC-FLAGS                         PIC S9(9) BINARY.
** Ver:1 **
** Number of milliseconds before reconnect attempt
 15  MQCBC-RECONNECTDELAY PIC S9(9) BINARY.
** Ver:2 **
```

Deklaracja języka PL/I dla MQCBC

```
dcl
 1 MQCBC based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version */
  3 CallType            fixed bin(31), /* Callback type */
  3 Hobj                fixed bin(31), /* Object Handle */
  3 CallbackArea        pointer,       /* User area passed to the function */
  3 ConnectionArea      pointer,       /* Connection User Area */
  3 CompCode            fixed bin(31); /* Completion Code */
  3 Reason              fixed bin(31); /* Reason Code */
  3 State               fixed bin(31); /* Consumer State */
  3 DataLength          fixed bin(31); /* Message Data Length */
  3 BufferLength        fixed bin(31); /* Message Buffer length */
  3 Flags               fixed bin(31); /* Consumer Flags */
/* Ver:1 */
  3 ReconnectDelay      fixed bin(31); /* Number of milliseconds before */
/* Ver:2 */                            /* reconnect attempt */
```

Deklaracja High Level Assembler dla MQCBC

```
    MQCBC                 DSECT
    MQCBC                 DS  0F    Force fullword alignment
    MQCBC_STRUCID         DS  CL4   Structure identifier
    MQCBC_VERSION         DS  F     Structure version number
    MQCBC_CALLTYPE        DS  F     Why Function was called
    MQCBC_HOBJ            DS  F     Object Handle
    MQCBC_CALLBACKAREA    DS  A     Callback data passed to the function
    MQCBC_CONNECTIONAREA  DS  A     MQCTL Data area passed to the function
    MQCBC_COMPCODE        DS  F     Completion Code
    MQCBC_REASON          DS  F     Reason Code
    MQCBC_STATE           DS  F     Consumer State
    MQCBC_DATALENGTH      DS  F     Message Data Length
    MQCBC_BUFFERLENGTH    DS  F     Buffer Length
    MQCBC_FLAGS           DS  F     Flags containing information about this consumer
    MQCBC_RECONNECTDELAY  DS  F     Number of milliseconds before reconnect
    MQCBC_LENGTH          EQU *-MQCBC
                          ORG       MQCBC
    MQCBC_AREA            DS  CL(MQCBC_LENGTH)
```

StrucId (MQCHAR4) dla MQCBC

Jest to identyfikator struktury struktury kontekstu wywołania zwrotnego. Jest to zawsze pole wejściowe. Jego wartością jest MQCBC_STRUC_ID.

Wartość musi być następująca:

MQCBC_ID_struktury
Identyfikator struktury kontekstu wywołania zwrotnego.

Dla języka programowania C zdefiniowana jest również stała MQCBC_STRUC_ID_ARRAY. Ta wartość jest taka sama jak wartość MQCBC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQCBC

Jest to numer wersji struktury kontekstu wywołania zwrotnego. Jest to zawsze pole wejściowe.

Wartość musi być następująca:

MQCBC_VERSION_1
Struktura kontekstu wywołania zwrotnego w wersji 1.

Następująca stała określa numer wersji bieżącej:

MQCBC_CURRENT_VERSION
Bieżąca wersja struktury kontekstu wywołania zwrotnego.

Do funkcji zwrotnej zawsze jest przekazywana najnowsza wersja struktury.

CallType (MQLONG) dla MQCBC

Pole zawierające informacje o przyczynie wywołania tej funkcji; zdefiniowane są następujące wartości.

Typy wywołań dostarczania komunikatów: te typy wywołań zawierają informacje o komunikacie. Parametry **DataLength** i **BufferLength** są poprawne dla tych typów wywołań.

MQCBCT_MSG_REMOVED
Funkcja konsumenta komunikatów została wywołana z komunikatem, który został destrukcyjnie usunięty z uchwytu obiektu.

Jeśli wartością *CompCode* jest MQCC_WARNING, wartością pola *Reason* jest MQRC_TRUNCATED_MSG_ACCEPTED lub jeden z kodów wskazujących na problem z konwersją danych.

MQCBCT_MSG_NOT_REMOVED (komunikat MQCBCT_NOT_REMOVED)
Funkcja konsumenta komunikatów została wywołana z komunikatem, który nie został jeszcze destrukcyjnie usunięty z uchwytu obiektu. Komunikat można destrukcyjnie usunąć z uchwytu obiektu za pomocą *MsgToken*.

Komunikat mógł nie zostać usunięty, ponieważ:

- Opcje MQGMO zażądały operacji przeglądania, MQGMO_BROWSE_ *
- Komunikat jest większy niż dostępny bufor, a opcje MQGMO nie określają parametru MQGMO_ACCEPT_TRUNCATED_MSG

Jeśli wartością parametru *CompCode* jest MQCC_WARNING, wartością pola *Reason* jest MQRC_TRUNCATED_MSG_FAILED lub jeden z kodów wskazujących na problem z konwersją danych.

Typy wywołań sterujących wywołania zwrotnego: te typy wywołań zawierają informacje o sterowaniu wywołaniem zwrotnym i nie zawierają szczegółów dotyczących komunikatu. Te typy wywołań są żądane przy użyciu opcji Opcje w strukturze MQCBD.

Parametry **DataLength** i **BufferLength** nie są poprawne dla tych typów wywołań.

WYWOŁANIE MQCBCT_REGISTER_CALL
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania początkowej konfiguracji.

Funkcja zwrotna jest wywoływana natychmiast po zarejestrowaniu wywołania zwrotnego, czyli po powrocie z wywołania MQCB przy użyciu wartości pola *Operation* w tabeli MQOP_REGISTER.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i dla procedur obsługi zdarzeń.

Na żądanie jest to pierwsze wywołanie funkcji zwrotnej.

Wartością pola *Reason* jest MQRC_NONE.

MQCBCT_START_CALL
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania pewnych czynności konfiguracyjnych podczas uruchamiania, na przykład przywrócenie zasobów, które zostały wyczyszczone podczas poprzedniego zatrzymania.

Funkcja zwrotna jest wywoływana, gdy połączenie jest uruchamiane za pomocą komendy MQOP_START lub MQOP_START_WAIT.

Jeśli funkcja zwrotna jest zarejestrowana w innej funkcji zwrotnej, ten typ wywołania jest wywoływany po powrocie z wywołania zwrotnego.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartością pola *Reason* jest MQRC_NONE.

MQCBCT_STOP_CALL
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania procedury czyszczącej, gdy zostanie ona zatrzymana na jakiś czas, na przykład w celu wyczyszczenia dodatkowych zasobów, które zostały uzyskane podczas odbierania komunikatów.

Funkcja zwrotna jest wywoływana, gdy wywołanie MQCTL zostanie wykonane przy użyciu wartości pola *Operation* komendy MQOP_STOP.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartość w polu *Reason* jest ustawiona w celu wskazania przyczyny zatrzymania.

WYWOŁANIA MQCBCT_DEREGISTER_CALL
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania końcowego czyszczenia po zakończeniu procesu konsumowania. Funkcja zwrotna jest wywoływana, gdy:

- Funkcja zwrotna jest wyrejestrowywana przy użyciu wywołania MQCB z MQOP_DEREGISTER.
- Kolejka jest zamknięta, co powoduje niejawne wyrejestrowanie. W tym przypadku do funkcji zwrotnej jako uchwyt obiektu przekazywany jest parametr MQHO_UNUSABLE_HOBJ.
- Wywołanie MQDISC zostało zakończone-spowodowało niejawne zamknięcie i wyrejestrowanie. W takim przypadku połączenie nie jest rozłączane natychmiast, a żadna transakcja w toku nie jest jeszcze zatwierdzona.

Jeśli dowolne z tych działań zostanie wykonane wewnątrz samej funkcji zwrotnej, działanie zostanie wywołane po powrocie z wywołania zwrotnego.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i dla procedur obsługi zdarzeń.

Na żądanie jest to ostatnie wywołanie funkcji zwrotnej.

Wartość w polu *Reason* jest ustawiona w celu wskazania przyczyny zatrzymania.

MQCBCT_EVENT_CALL
Funkcja procedury obsługi zdarzeń

Funkcja procedury obsługi zdarzeń została wywołana bez komunikatu, gdy menedżer kolejek lub połączenie zostało zatrzymane lub wyciszone.

Tego wywołania można użyć do podjęcia odpowiednich działań dla wszystkich funkcji zwrotnych.

Funkcja konsumenta komunikatów

Funkcja konsumenta komunikatów została wywołana bez komunikatu w przypadku wykrycia błędu (*CompCode* = MQCC_FAILED), który jest specyficzny dla uchwytu obiektu, na przykład *Reason* code = MQRC_GET_INHIBITED.

Wartość w polu *Reason* jest ustawiona w celu wskazania przyczyny wywołania.

MQCBCT_MC_EVENT_CALL

Funkcja procedury obsługi zdarzeń została wywołana dla zdarzeń rozsyłania grupowego. Procedura obsługi zdarzeń jest wysyłana jako IBM MQ zdarzenia rozsyłania grupowego zamiast zwykłych IBM MQ zdarzeń.

Więcej informacji na temat wywołania MQCBCT_MC_EVENT_CALL zawiera sekcja Raportowanie wyjątków rozsyłania.

Hobj (MQHOBJ) dla MQCBC

Jest to uchwyt obiektu dla wywołań do konsumenta komunikatów.

W przypadku procedury obsługi zdarzeń ta wartość to MQHO_NONE

Aplikacja może użyć tego uchwytu i znacznika komunikatu w bloku Opcje pobierania komunikatu, aby pobrać komunikat, jeśli komunikat nie został usunięty z kolejki.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQHO_UNUSABLE_HOBJ.

CallbackArea (MQPTR) dla MQCBC

To pole jest dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji w oparciu o treść tego pola i jest ono przekazywane bez zmian z pola CallbackArea w strukturze MQCBD, które jest parametrem wywołania MQCB używanym do definiowania funkcji zwrotnej.

Zmiany wprowadzone w pliku *CallbackArea* są zachowywane w wywołaniach funkcji zwrotnej dla *HObj*. To pole nie jest współużytkowane z funkcjami zwrotnymi dla innych uchwytów.

Jest to pole wejściowe/wyjściowe funkcji zwrotnej. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

ConnectionArea (MQPTR) dla MQCBC

To pole jest dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest ono przekazywane bez zmian z pola ConnectionArea w strukturze MQCTLO, które jest parametrem wywołania MQCTL używanym do sterowania funkcją zwrotną.

Wszelkie zmiany wprowadzone w tym polu przez funkcje zwrotne są zachowywane w wywołaniach funkcji zwrotnej. Ten obszar może być używany do przekazywania informacji, które mają być współużytkowane przez wszystkie funkcje zwrotne. W przeciwieństwie do obszaru *CallbackArea*, ten obszar jest wspólny dla wszystkich wywołań zwrotnych uchwytu połączenia.

Jest to pole wejściowe i wyjściowe. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

CompCode (MQLONG) dla MQCBC

To pole zawiera kod zakończenia. Wskazuje, czy wystąpiły problemy podczas odbierania komunikatu.

Jest to jedna z następujących wartości:

MQCC_OK
Pomyślne zakończenie

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Jest to pole wejściowe. Wartością początkową tego pola jest MQCC_OK.

Przyczyna (MQLONG) dla MQCBC
Jest to kod przyczyny określający *CompCode*.

Jest to pole wejściowe. Wartością początkową tego pola jest MQRC_NONE.

Stan (MQLONG) dla MQCBC
Wskazanie stanu bieżącego konsumenta. To pole jest najbardziej wartościowe dla aplikacji, gdy do funkcji konsumenta przekazywany jest niezerowy kod przyczyny.

Tego pola można użyć, aby uprościć programowanie aplikacji, ponieważ nie ma potrzeby kodowania zachowania dla każdego kodu przyczyny.

Jest to pole wejściowe. Wartością początkową tego pola jest MQCS_NONE.

Tabela 472.

Stan	**Działanie menedżera kolejek**	**Wartość stałej**
MQCS_NONE Ten kod przyczyny reprezentuje normalne wywołanie bez dodatkowych informacji o przyczynie.	Brak; jest to normalne działanie.	0
MQCS_SUSPENDED_TEMPORARY Te kody przyczyny reprezentują warunki tymczasowe.	Procedura zwrotna jest wywoływana w celu zgłoszenia warunku, a następnie zawieszona. Po pewnym czasie system może ponowić operację, co może spowodować ponowne wystąpienie tego samego warunku.	1
MQCS_SUSPENDED_USER_ACTION Te kody przyczyny reprezentują warunki, w których wywołanie zwrotne musi podjąć działanie w celu rozwiązania warunku.	Konsument jest zawieszony, a procedura wywołania zwrotnego jest wywoływana w celu zgłoszenia warunku. Procedura wywołania zwrotnego powinna rozwiązać warunek, jeśli to możliwe, i albo RESUME, albo zamknąć połączenie.	2
MQCS_SUSPENDED Te kody przyczyny reprezentują niepowodzenia, które uniemożliwiają dalsze wywołania zwrotne komunikatów.	Menedżer kolejek automatycznie zawiesza funkcję zwrotną. Jeśli funkcja zwrotna zostanie wznowiona, prawdopodobnie ponownie otrzyma ten sam kod przyczyny.	3
MQCS_STOPPED Te kody przyczyny reprezentują koniec wykorzystania komunikatów.	Dostarczone do procedury obsługi wyjątków i do wywołań zwrotnych, które określały MQCBDO_STOP_CALL. Nie można korzystać z kolejnych komunikatów.	4

DataLength (MQLONG) dla MQCBC
Jest to długość (w bajtach) danych aplikacji w komunikacie. Jeśli wartość wynosi zero, oznacza to, że komunikat nie zawiera danych aplikacji.

Pole DataLength zawiera długość komunikatu, ale niekoniecznie długość danych komunikatu przekazanych do konsumenta. Możliwe, że komunikat został obcięty. Pole ReturnedLength w obiekcie MQGMO służy do określenia, ile danych zostało faktycznie przekazanych konsumentowi.

Jeśli kod przyczyny wskazuje, że komunikat został obcięty, można użyć pola DataLength , aby określić wielkość rzeczywistego komunikatu. Umożliwia to określenie wielkości buforu wymaganego do

przechowywania danych komunikatu, a następnie wywołanie MQCB w celu zaktualizowania parametru MaxMsgLength przy użyciu odpowiedniej wartości.

Jeśli określono opcję MQGMO_CONVERT, przekształcony komunikat może być większy niż wartość zwrócona dla DataLength. W takich przypadkach aplikacja prawdopodobnie musi wydać wywołanie MQCB w celu zaktualizowania parametru MaxMsgLength na wartość większą niż wartość zwrócona przez menedżer kolejek dla parametru DataLength.

Aby uniknąć problemów z obcinaniem komunikatów, należy określić wartość MaxMsgLength jako MQCBD_FULL_MSG_LENGTH. Powoduje to, że menedżer kolejek przydziela bufor dla pełnej długości komunikatu po konwersji danych. Należy jednak pamiętać, że nawet jeśli ta opcja jest określona, nadal możliwe jest, że nie jest dostępna wystarczająca ilość pamięci do poprawnego przetworzenia żądania. Aplikacje powinny zawsze sprawdzać zwrócony kod przyczyny. Jeśli na przykład nie można przydzielić wystarczającej ilości pamięci do przekształcenia komunikatu, komunikaty są zwracane do aplikacji bez konwersji.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

BufferLength (MQLONG) dla MQCBC

To pole jest długością (w bajtach) buforu komunikatów, który został przekazany do tej funkcji.

Bufor może być większy niż zarówno wartość MaxMsgLength zdefiniowana dla konsumenta, jak i wartość ReturnedLength w obiekcie MQGMO.

Rzeczywista długość komunikatu jest podawana w polu DataLength .

Aplikacja może używać całego buforu do własnych celów w czasie trwania funkcji zwrotnej.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi wyjątku.

Flagi (MQLONG) dla MQCBC

Flagi zawierające informacje o tym konsumencie.

Zdefiniowana jest następująca opcja:

MQCBCF_READA_BUFFER_EMPTY

Ta opcja może zostać zwrócona, jeśli poprzednie wywołanie MQCLOSE z opcją MQCO_QUIESCE nie powiodło się z kodem przyczyny MQRC_READ_AHEAD_MSGS.

Ten kod wskazuje, że jest zwracany ostatni komunikat odczytu z wyprzedzeniem i że bufor jest teraz pusty. Jeśli aplikacja wysyła kolejne wywołanie MQCLOSE przy użyciu opcji MQCO_QUIESCE), zakończy się powodzeniem.

Należy zauważyć, że nie ma gwarancji, że aplikacja otrzyma komunikat z tym zestawem flag, ponieważ w buforze odczytu z wyprzedzeniem mogą nadal znajdować się komunikaty, które nie są zgodne z bieżącymi kryteriami wyboru. W tym przypadku funkcja konsumenta jest wywoływana z kodem przyczyny MQRC_HOBJ_QUIESCED.

Jeśli bufor odczytu z wyprzedzeniem jest całkowicie pusty, konsument jest wywoływany z flagą MQCBCF_READA_BUFFER_EMPTY i kodem przyczyny MQRC_HOBJ_QUIESCED_NO_MSGS.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

ReconnectDelay (MQLONG) dla MQCBC

Parametr ReconnectDelay wskazuje czas oczekiwania menedżera kolejek przed podjęciem próby ponownego nawiązania połączenia. Pole może być modyfikowane przez procedurę obsługi zdarzeń w celu zmiany opóźnienia lub zatrzymania ponownego połączenia.

Pola `ReconnectDelay` należy używać tylko wtedy, gdy wartością pola `Przyczyna` w kontekście wywołania zwrotnego jest `MQRC_RECONNECTING`.

W momencie wejścia do procedury obsługi zdarzeń wartość parametru `ReconnectDelay` jest liczbą milisekund, przez które menedżer kolejek ma oczekiwać przed podjęciem próby ponownego nawiązania połączenia. Tabela 473 na stronie 293 zawiera listę wartości, które można ustawić w celu zmodyfikowania zachowania menedżera kolejek w przypadku powrotu z procedury obsługi zdarzeń.

Tabela 473. `ReconnectDelay` wartości

Nazwa	Wartość	Opis
`MQRD_NO_RECONNECT`	`-1`	Nie wykonuj więcej prób ponownego połączenia. Do aplikacji jest zwracany błąd.
`MQRD_NO_DELAY`	`0`	Spróbuj połączyć się ponownie natychmiast.
`Milliseconds`	`>0`	Przed ponowną próbą nawiązania połączenia odczekaj ten czas (w milisekundach).

MQCBD-deskryptor wywołania zwrotnego

Struktura MQCBD służy do określania funkcji zwrotnej i opcji sterujących jej użyciem przez menedżer kolejek. Struktura jest parametrem wejściowym wywołania MQCB.

Dostępność

Struktura MQCBD jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Wersja

Bieżąca wersja MQCBD to MQCBD_VERSION_1.

Zestaw znaków i kodowanie

Dane w strukturze MQCBD muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 474. Pola w MQCBD

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucID (identyfikator struktury)	MQCBD_STRUC_ID (Identyfikator struktury CBC)	'CBD¬'
Wersja (numer wersji struktury)	MQCBD_VERSION_1	1
CallbackType (typ funkcji zwrotnej)	MQCBT_MESSAGE_CONSUMER	1
Opcje (opcje sterujące zużyciem komunikatów)	MQCBDO_BRAK	0
CallbackArea (pole używane przez funkcję zwrotną)	Brak	Pusty wskaźnik lub puste znaki
CallbackFunction (określa, czy funkcja jest wywoływana jako wywołanie interfejsu API)	Brak	Pusty wskaźnik lub puste znaki
CallbackName (określa, czy funkcja jest wywoływana jako program dowiązany dynamicznie)	Brak	Pusty łańcuch lub odstępy
MaxMsgDługość (długość najdłuższego komunikatu, jaki można odczytać)	MQCBD_FULL_MSG_LENGTH	-1

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub spacje oznacza zerową wartość w języku programowania C i puste znaki w innych językach programowania.
3. W języku programowania C: zmienna makraMQCBD_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCBD MyCBD = {MQCBD_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQCBD

```
typedef struct tagMQCBD MQCBD;
 struct tagMQCBD {
   MQCHAR4    StrucId;                /* Structure identifier */
   MQLONG     Version;                /* Structure version number */
   MQLONG     CallBackType;           /* Callback function type */
   MQLONG     Options;                /* Options controlling message
                                         consumption */
   MQPTR      CallbackArea;           /* User data passed to the function */
   MQPTR      CallbackFunction;       /* Callback function pointer */
   MQCHAR128  CallbackName;           /* Callback name */
   MQLONG     MaxMsgLength;           /* Maximum message length */
 };
```

Deklaracja języka COBOL dla MQCBD

```
** MQCBCD structure
 10  MQCBD.
** Structure Identifier
 15  MQCBD-STRUCID                    PIC X(4).
** Structure Version
 15  MQCBD-VERSION                    PIC S9(9) BINARY.
```

```
 ** Callback Type
  15  MQCBD-CALLBACKTYPE                 PIC S9(9) BINARY.
** Options
  15  MQCBD-OPTIONS                      PIC S9(9) BINARY.
** Callback User Area
  15  MQCBD-CALLBACKAREA                 POINTER
** Callback Function Pointer
  15  MQCBD-CALLBACKFUNCTION             FUNCTION-POINTER
** Callback Program Name
  15  MQCBD-CALLBACKNAME                 PIC X(128)
** Maximum Message Length
  15  MQCDB-MAXMSGLENGTH                 PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQCBD

```
dcl
 1 MQCBD based,
   3 StrucId             char(4),       /* Structure identifier*/
   3 Version             fixed bin(31), /* Structure version*/
   3 CallbackType        fixed bin(31), /* Callback function type */
   3 Options             fixed bin(31), /* Options */
   3 CallbackArea        pointer,       /* User area passed to the function */
   3 CallbackFunction    pointer,       /* Callback Function Pointer */
   3 CallbackName        char(128),     /* Callback Program Name */
   3 MaxMsgLength        fixed bin(31); /* Maximum Message Length */
```

StrucId (MQCHAR4) dla MQCBD

Jest to identyfikator struktury struktury deskryptora wywołania zwrotnego. Jest to zawsze pole wejściowe. Jego wartością jest MQCBD_STRUC_ID.

Wartość musi być następująca:

MQCBD_STRUC_ID (Identyfikator struktury CBC)
Identyfikator struktury deskryptora wywołania zwrotnego.

Dla języka programowania C zdefiniowana jest również stała MQCBD_STRUC_ID_ARRAY. Ma taką samą wartość jak MQCBD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQCBD

Jest to numer wersji struktury deskryptora wywołania zwrotnego. Jest to zawsze pole wejściowe.

Wartość musi być następująca:

MQCBD_VERSION_1
Struktura deskryptora wywołania zwrotnego w wersji 1.

Następująca stała określa numer wersji bieżącej:

MQCBD_CURRENT_VERSION
Bieżąca wersja struktury deskryptora wywołania zwrotnego.

CallbackType (MQLONG) dla MQCBD

Struktura deskryptora wywołania zwrotnego-pole CallbackType

Jest to typ funkcji zwrotnej. Wartość musi być jedną z następujących wartości:

MQCBT_MESSAGE_CONSUMER
Definiuje wywołanie zwrotne jako funkcję konsumenta komunikatów.

Funkcja zwrotna konsumenta komunikatów jest wywoływana, gdy komunikat spełniający określone kryteria wyboru jest dostępny w uchwycie obiektu i połączenie jest uruchamiane.

MQCBT_EVENT_HANDLER
Definiuje to wywołanie zwrotne jako procedurę zdarzenia asynchronicznego; nie jest sterowane odbieraniem komunikatów dla uchwytu.

Parametr *Hobj* nie jest wymagany w wywołaniu obiektu MQCB definiującym procedurę obsługi zdarzeń i jest ignorowany, jeśli został określony.

Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko konsumenta komunikatów. Funkcja konsumenta jest wywoływana bez komunikatu, gdy wystąpi zdarzenie, na przykład zatrzymanie menedżera kolejek lub połączenia albo wyciszenie. Nie jest on wywoływany dla warunków specyficznych dla pojedynczego konsumenta komunikatów, na przykład MQRC_GET_INHIBITED.

Zdarzenia są dostarczane do aplikacji, niezależnie od tego, czy połączenie jest uruchomione, czy zatrzymane, z wyjątkiem następujących środowisk:

- Środowisko CICS w systemie z/OS
- aplikacje niewielowątkowe

Jeśli program wywołujący nie przekaże jednej z tych wartości, wywołanie zakończy się niepowodzeniem z kodem *Reason* MQRC_CALLBACK_TYPE_ERROR.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQCBT_MESSAGE_CONSUMER.

Opcje (MQLONG) dla MQCBD

Struktura deskryptora wywołania zwrotnego-pole Opcje

Można określić jedną lub więcej z tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

MQCBDO_FAIL_IF_QUIESCING,
Wywołanie MQCB kończy się niepowodzeniem, jeśli menedżer kolejek jest w stanie wyciszania.

W systemie z/OSta opcja wymusza także niepowodzenie wywołania MQCB, jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wyciszania.

Określ opcję MQGMO_FAIL_IF_QUIESCING w opcjach MQGMO przekazywanych w wywołaniu MQCB, aby spowodować wysłanie powiadomienia do konsumentów komunikatów podczas wyciszania.

Opcje sterowania: Następujące opcje sterują tym, czy funkcja zwrotna jest wywoływana bez komunikatu, gdy zmienia się stan konsumenta:

MQCBDO_REGISTER_CALL,
Funkcja zwrotna jest wywoływana z typem wywołania MQCBCT_REGISTER_CALL.

MQCBDO_START_CALL
Funkcja zwrotna jest wywoływana z typem wywołania MQCBCT_START_CALL.

MQCBDO_STOP_CALL
Funkcja zwrotna jest wywoływana z typem wywołania MQCBCT_STOP_CALL.

WYWOŁANIA MQCBDO_DEREGISTER_CALL
Funkcja zwrotna jest wywoływana z typem wywołania MQCBCT_DEREGISTER_CALL.

WYWOŁANIA MQCBDO_EVENT_CALL
Funkcja zwrotna jest wywoływana z typem wywołania MQCBCT_EVENT_CALL.

MQCBDO_MC_EVENT_CALL
Funkcja zwrotna jest wywoływana z typem wywołania MQCBCT__MC_EVENT_CALL.

Więcej informacji na temat tych typów wywołań zawiera sekcja CallType .

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

MQCBDO_BRAK
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

Parametr MQCBDO_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Opcja ta nie powinna być używana z żadną inną opcją, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Jest to pole wejściowe. Wartością początkową pola *Options* jest MQCBDO_NONE.

CallbackArea (MQPTR) dla MQCBD

Struktura deskryptora wywołania zwrotnego-pole CallbackArea

Jest to pole dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest ono przekazywane bez zmian z pola CallbackArea w strukturze MQCBC, która jest parametrem deklaracji funkcji zwrotnej.

Ta wartość jest używana tylko w przypadku produktu *Operation* , który ma wartość MQOP_REGISTER i nie ma obecnie zdefiniowanego wywołania zwrotnego. Nie zastępuje ona poprzedniej definicji.

Jest to pole wejściowe i wyjściowe funkcji zwrotnej. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

CallbackFunction (MQPTR) dla MQCBD

Struktura deskryptora wywołania zwrotnego-pole CallbackFunction

Funkcja zwrotna jest wywoływana jako wywołanie funkcji.

To pole służy do określania wskaźnika do funkcji zwrotnej.

Należy podać wartość *CallbackFunction* lub *CallbackName*. W przypadku podania obu wartości zostanie zwrócony kod przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Jeśli nie ustawiono wartości *CallbackName* ani *CallbackFunction* , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Ta opcja nie jest obsługiwana w następującym środowisku: Języki programowania i kompilatory, które nie obsługują odwołań do wskaźników funkcji. W takich sytuacjach wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

z/OS W systemie z/OSfunkcja musi być wywoływana z konwencjami łączenia systemu operacyjnego. Na przykład w języku programowania C należy określić:

```
#pragma linkage(MQCB_FUNCTION,OS)
```

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

Uwaga: Jeśli produkt CICS jest używany z produktem IBM WebSphere MQ 7.0.1, wykorzystanie asynchroniczne jest obsługiwane, jeśli:

- Poprawka APAR PK66866 jest stosowana do systemu CICS TS 3.2
- Apar PK89844 jest stosowany do CICS TS 4.1

CallbackName (MQCHAR128) dla MQCBD

Struktura deskryptora wywołania zwrotnego-pole CallbackName

Funkcja zwrotna jest wywoływana jako program dowiązany dynamicznie.

Należy podać wartość *CallbackFunction* lub *CallbackName*. W przypadku podania obu wartości zostanie zwrócony kod przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Jeśli nie ustawiono wartości *CallbackName* ani *CallbackFunction* , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CALLBACK_ROUTINE_ERROR.

Moduł jest ładowany, gdy rejestrowana jest pierwsza procedura zwrotna, która ma zostać użyta, i rozładowywany, gdy ostatnia procedura zwrotna, która ma go użyć, wyrejestrowuje.

Z wyjątkiem przypadków opisanych w poniższym tekście, nazwa jest wyrównywana do lewej strony w polu bez odstępów wewnętrznych; sama nazwa jest dopełniana odstępami do długości pola. W poniższych opisach nawiasy kwadratowe ([]) oznaczają informacje opcjonalne:

IBM i

Nazwa wywołania zwrotnego może mieć jeden z następujących formatów:

- Biblioteka "/" Program
- Biblioteka "/" ServiceProgram "("FunctionName")"

Na przykład: `MyLibrary/MyProgram(MyFunction)`.

Nazwą biblioteki może być *LIBL. Nazwy biblioteki i programu są ograniczone do maksymalnie 10 znaków.

AIX and Linux
Nazwa wywołania zwrotnego jest nazwą dynamicznie ładowanego modułu lub biblioteki z przyrostkiem nazwy funkcji rezydującej w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu:

```
[path]library(function)
```

Jeśli ścieżka nie zostanie podana, zostanie użyta systemowa ścieżka wyszukiwania.

Długość nazwy nie może przekraczać 128 znaków.

Windows
Nazwa wywołania zwrotnego jest nazwą biblioteki dołączanej dynamicznie z przyrostkiem nazwy funkcji rezydującej w tej bibliotece. Nazwa funkcji musi być ujęta w nawias. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu i napędem:

```
[d:][path]library(function)
```

Jeśli napęd i ścieżka nie są określone, używana jest systemowa ścieżka wyszukiwania.

Długość nazwy nie może przekraczać 128 znaków.

z/OS
Nazwa wywołania zwrotnego jest nazwą modułu ładującego, która jest poprawna dla specyfikacji parametru EP makra LINK lub LOAD.

Długość nazwy nie może przekraczać 8 znaków.

z/OS CICS
Nazwa wywołania zwrotnego jest nazwą modułu ładującego, która jest poprawna dla specyfikacji parametru PROGRAM makra komendy EXEC CICS LINK.

Długość nazwy nie może przekraczać 8 znaków.

Program można zdefiniować jako zdalny za pomocą opcji REMOTESYTEM zainstalowanej definicji PROGRAM lub za pomocą programu routingu dynamicznego.

Zdalny region CICS musi być połączony z systemem IBM MQ , jeśli program ma używać wywołań funkcji API języka IBM MQ . Należy jednak zauważyć, że pole Hobj w strukturze MQCBC nie jest poprawne w systemie zdalnym.

Jeśli wystąpi błąd podczas próby załadowania pliku *CallbackName*, do aplikacji zwracany jest jeden z następujących kodów błędu:

- MQRC_MODULE_NOT_FOUND
- MQRC_MODULE_INVALID
- MQRC_MODULE_ENTRY_NOT_FOUND

W dzienniku błędów zapisywany jest również komunikat zawierający nazwę modułu, dla którego próbowano wykonać ładowanie, oraz kod przyczyny niepowodzenia z systemu operacyjnego.

Jest to pole wejściowe. Wartością początkową tego pola jest łańcuch pusty lub pusty łańcuch.

MaxMsg(MQLONG) dla MQCBD
Jest to długość (w bajtach) najdłuższego komunikatu, który można odczytać z uchwytu i przekazać do procedury zwrotnej. Struktura deskryptora wywołania zwrotnego-pole MaxMsgDługość

Jeśli komunikat ma dłuższą długość, procedura zwrotna odbiera *MaxMsgLength* bajtów komunikatu i kod przyczyny:

- MQRC_TRUNCATED_MSG_FAILED lub
- MQRC_TRUNCATED_MSG_ACCEPTED, jeśli określono MQGMO_ACCEPT_TRUNCATED_MSG.

Rzeczywista długość komunikatu jest podawana w polu DataLength struktury MQCBC.

Zdefiniowana jest następująca wartość specjalna:

MQCBD_FULL_MSG_LENGTH
Długość buforu jest dopasowywana przez system w taki sposób, aby komunikaty były zwracane bez obcinania.

Jeśli ilość dostępnej pamięci jest niewystarczająca do przydzielenia buforu w celu odebrania komunikatu, system wywołuje funkcję zwrotną z kodem przyczyny MQRC_STORAGE_NOT_AVAILABLE.

Jeśli na przykład żądana jest konwersja danych, a ilość dostępnej pamięci jest niewystarczająca do przekształcenia danych komunikatu, nieprzekształcony komunikat jest przekazywany do funkcji zwrotnej.

Jest to pole wejściowe. Wartością początkową pola *MaxMsgLength* jest MQCBD_FULL_MSG_LENGTH.

MQCHARV-łańcuch o zmiennej długości

Struktura MQCHARV służy do opisu łańcucha o zmiennej długości.

Dostępność

Struktura MQCHARV jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

Zestaw znaków i kodowanie

Dane w tabeli MQCHARV muszą być zakodowane w menedżerze kolejek lokalnych, który jest nadawany przez zmienną MQENC_NATIVE i zestaw znaków pola VSCCSID w strukturze. Jeśli aplikacja działa jako klient produktu MQ , struktura musi być zakodowana w kliencie. Niektóre zestawy znaków mają reprezentację zależną od kodowania. Jeśli VSCCSID jest jednym z tych zestawów znaków, użyte kodowanie jest takie samo jak kodowanie innych pól w MQCHARV. Zestaw znaków identyfikowany przez VSCCSID może być zestawem znaków dwubajtowych (DBCS).

Użycie

Struktura MQCHARV odnosi się do danych, które mogą być nieciągłe w strukturze zawierającej te dane. W celu adresowania tych danych można użyć pól zadeklarowanych za pomocą typu danych wskaźnika. Należy pamiętać, że język COBOL nie obsługuje typu danych wskaźnika we wszystkich środowiskach. Z tego powodu dane mogą być również adresowane przy użyciu pól, które zawierają przesunięcie danych od początku struktury zawierającej obiekt MQCHARV.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 475. Pola w tabeli MQCHARV

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
VSPtr (wskaźnik do łańcucha o zmiennej długości)	Brak	Wskaźnik pusty lub bajty puste.
VSOffset (przesunięcie w bajtach łańcucha o zmiennej długości od początku struktury zawierającej tę strukturę MQCHARV)	Brak	0
VSBufSize (wielkość w bajtach buforu adresowanego przez pole VSPtr lub VSOffset)	MQVS_USE_VSLENGTH (mechanizm MQVS)	0
VSLength (długość w bajtach łańcucha o zmiennej długości adresowanego przez pole VSPtr lub VSOffset)	Brak	0
VSCCSID (identyfikator zestawu znaków łańcucha o zmiennej długości, adresowanego przez pole VSPtr lub VSOffset)	APPL MQCCSI_PL	-3

Uwaga: W języku programowania C zmienna makra MQCHARV_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCHARV MyVarStr = {MQCHARV_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQCHARV

```
typedef struct tagMQCHARV MQCHARV;
struct tagMQCHARV {
  MQPTR     VSPtr;               /* Address of variable length string */
  MQLONG    VSOffset;            /* Offset of variable length string */
  MQLONG    VSBufSize;           /* Size of buffer */
  MQLONG    VSLength;            /* Length of variable length string */
  MQLONG    VSCCSID;             /* CCSID of variable length string */
};
```

Deklaracja języka COBOL dla 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.
```

Uwaga: Jeśli aplikacje w języku COBOL mają być przesyłane między środowiskami, należy sprawdzić, czy typ danych wskaźnika jest dostępny we wszystkich planowanych środowiskach. Jeśli nie, aplikacja musi adresować dane przy użyciu pól przesunięcia zamiast pól wskaźnika. W środowiskach, w których wskaźniki nie są obsługiwane, można zadeklarować pola wskaźników jako łańcuchy bajtowe o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null. Nie należy zmieniać tej wartości początkowej, jeśli używane są pola przesunięcia. Jednym ze sposobów na to, aby to zrobić bez zmiany dostarczonych książeczek kopii, jest użycie następujących:

```
COPY CMQCHRVV REPLACING POINTER BY ==BINARY PIC S9(9)==.
```

gdzie można wymienić CMQCHRVV na książkę kopii, która ma być używana.

Deklaracja PL/I dla MQCHARV

```
dcl
 1 MQCHARV based,
  3 VSPtr       pointer,       /* Address of variable length string */
  3 VSOffset    fixed bin(31), /* Offset of variable length string */
  3 VSBufSize   fixed bin(31), /* Size of buffer */
  3 VSLength    fixed bin(31), /* Length of variable length string */
  3 VSCCSID     fixed bin(31); /* CCSID of variable length string */
```

Deklaracja High Level Assembler dla MQCHARV

```
MQCHARV             DSECT
MQCHARV_VSPTR       DS   F     Address of variable length string
MQCHARV_VSOFFSET    DS   F     Offset of variable length string
MQCHARV_VSBUFSIZE   DS   F     Size of buffer
MQCHARV_VSLENGTH    DS   F     Length of variable length string
MQCHARV_VSCCSID     DS   F     CCSID of variable length string
*
MQCHARV_LENGTH      EQU  *-MQCHARV
                    ORG  MQCHARV
MQCHARV_AREA        DS   CL(MQCHARV_LENGTH)
```

VSPtr (MQPTR) dla MQCHARV

Jest to wskaźnik do łańcucha o zmiennej długości.

Do określenia łańcucha o zmiennej długości można użyć pola VSPtr lub VSOffset, ale nie obu.

Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

VSOffset (MQLONG) dla MQCHARV

Przesunięcie może być dodatnie lub ujemne. Do określenia łańcucha o zmiennej długości można użyć pola VSPtr lub VSOffset, ale nie obu. Przesunięcie (w bajtach) łańcucha o zmiennej długości od początku tabeli MQCHARV lub struktury zawierającej ten łańcuch.

Jeśli struktura MQCHARV jest osadzona w innej strukturze, ta wartość jest przesunięciem (w bajtach) łańcucha o zmiennej długości od początku struktury zawierającej tę strukturę MQCHARV. Jeśli struktura MQCHARV nie jest osadzona w innej strukturze, na przykład jeśli jest określona jako parametr w wywołaniu funkcji, przesunięcie jest względne w stosunku do początku struktury MQCHARV.

Wartością początkową tego pola jest 0.

VSBufSize (MQLONG) dla MQCHARV

Jest to wielkość (w bajtach) buforu adresowanego przez pole VSPtr lub VSOffset.

Jeśli struktura MQCHARV jest używana jako pole wyjściowe w wywołaniu funkcji, to pole musi być inicjowane z podaną długością buforu. Jeśli wartość VSLength jest większa niż VSBufSize , do programu wywołującego w buforze są zwracane tylko VSBufSize bajtów danych.

Ta wartość musi być wartością większą lub równą zero lub następującą wartością specjalną, która jest rozpoznawana:

MQVS_USE_VSLENGTH (mechanizm MQVS)
Po określeniu długość buforu jest pobierana z pola VSLength w strukturze MQCHARV. Nie należy używać tej wartości, jeśli struktura jest używana jako pole wyjściowe i podano bufor.

Jest to wartość początkowa tego pola.

Długość VSLength (MQLONG) dla MQCHARV

Długość (w bajtach) łańcucha o zmiennej długości, adresowanego przez pole VSPtr lub VSOffset.

Wartością początkową tego pola jest 0. Wartość musi być większa lub równa zero lub następująca wartość specjalna, która jest rozpoznawana:

MQVS_NULL_PRZERWANE

Jeśli parametr MQVS_NULL_TERMINATED nie zostanie określony, jako część łańcucha zostaną dołączone bajty VSLength. Jeśli występują znaki o kodzie zero, łańcuch nie jest ograniczany.

Jeśli określono opcję MQVS_NULL_TERMINATED, łańcuch jest ograniczony przez pierwszą wartość NULL napotkaną w łańcuchu. Sama wartość NULL nie jest uwzględniana jako część tego łańcucha.

Uwaga: Znak o kodzie zero używany do kończenia łańcucha, jeśli określono parametr MQVS_NULL_TERMINATED, jest znakiem o kodzie zero z zestawu kodowego określonego przez parametr VSCCSID.

Na przykład w UTF-16 (CCSID 1200, 13488 i 17584) jest to dwubajtowe kodowanie Unicode, w którym wartość null jest reprezentowana przez 16-bitową liczbę wszystkich zer. W UTF-16 powszechne jest wyszukiwanie pojedynczych bajtów ustawionych na zero, które są częścią znaków (na przykład 7-bitowych znaków ASCII), ale łańcuchy będą kończone znakiem o kodzie zero tylko wtedy, gdy na granicy parzystego bajtu zostaną znalezione dwa bajty o kodzie zero. Można uzyskać dwa 'zero' bajtów na granicy nieparzystej, gdy są one częścią poprawnych znaków. Na przykład x '01' x '00 x' 00 'x' 30 ' reprezentuje dwa poprawne znaki Unicode i nie kończy łańcucha znakiem o kodzie zero.

VSCCSID (MQLONG) dla MQCHARV

Jest to identyfikator zestawu znaków łańcucha o zmiennej długości, adresowany przez pole **VSPtr** lub **VSOffset** .

Wartość początkowa tego pola to *MQCCSI_APPL*, która jest definiowana przez produkt MQ w celu wskazania, że należy ją zmienić na rzeczywisty identyfikator zestawu znaków bieżącego procesu. W rezultacie wartość stałej *MQCCSI_APPL* nigdy nie jest powiązana z łańcuchem o zmiennej długości.

Początkową wartość tego pola można zmienić, definiując inną wartość stałej *MQCCSI_APPL* dla jednostki kompilacji. Sposób wykonania tej czynności zależy od języka programowania aplikacji.

z/OS W systemach z/OS domyślna aplikacja CCSID używana przez *MQCCSI_APPL* jest zdefiniowana w następujący sposób:

- W przypadku wsadowych aplikacji LE korzystających z interfejsu DLL wartością domyślną jest CODESET powiązana z bieżącymi ustawieniami narodowymi w momencie wydania komendy **MQCONN** (wartością domyślną jest 1047).
- W przypadku aplikacji wsadowych LE powiązanych z jednym z wsadowych kodów pośredniczących MQ wartością domyślną jest CODESET powiązana z bieżącymi ustawieniami narodowymi w czasie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku aplikacji wsadowych innych niż LE działających w wątku z/OS UNIX System Services wartością domyślną jest wartość THLICCSID w czasie pierwszego wywołania MQI wykonanego po wykonaniu funkcji **MQCONN** (wartością domyślną jest 1047).
- W przypadku innych aplikacji wsadowych wartością domyślną jest CCSID menedżera kolejek.

Ponowna definicja MQCCSI_APPL

W poniższych przykładach przedstawiono, w jaki sposób można przesłonić wartość MQCCSI_APPL w różnych językach programowania. Wartość parametru MQCCSI_APPL można zmienić, usuwając konieczność oddzielnego ustawiania identyfikatora VSCCSID dla każdego łańcucha o zmiennej długości. W tych przykładach identyfikator CCSID jest ustawiony na 1208; należy zmienić tę wartość na wymaganą. Staje się to wartością domyślną, którą można przesłonić, ustawiając identyfikator VSCCSID w dowolnej konkretnej instancji MQCHARV.

Składnia języka C

```
#define MQCCSI_APPL 1208
#include <cmqc.h>
```

Składnia języka COBOL

```
COPY CMQXYZV REPLACING -3 BY 1208.
```

Składnia języka PL/I

```
%MQCCSI_APPL = '1208';
%include syslib(cmqp);
```

Użycie High Level Assembler

```
MQCCSI_APPL EQU 1208
CMQA LIST=NO
```

MQCIH-nagłówek CICS bridge

Struktura MQCIH opisuje informacje nagłówka dla komunikatu wysyłanego do CICS przez CICS bridge.

W przypadku dowolnej platformy obsługiwanej przez produkt IBM MQ można utworzyć i przesłać komunikat zawierający strukturę MQCIH, ale tylko menedżer kolejek systemu IBM MQ for z/OS może używać parametru CICS bridge. Dlatego, aby komunikat dotarły do produktu CICS z menedżera kolejek innego niżz/OS , sieć menedżera kolejek musi zawierać co najmniej jeden menedżer kolejek produktu z/OS , przez który komunikat może być kierowany.

Wszystkie wersje produktu CICS obsługiwane przez produkt IBM MQ 9.0.0i nowsze używają wersji mostu dostarczonej przez firmę CICS . Więcej informacji na temat konfigurowania adaptera IBM MQ CICS i komponentów produktu IBM MQ CICS bridge zawiera sekcja Konfigurowanie połączeń z produktem MQ w dokumentacji produktu CICS .

Dostępność

Struktura MQCIH jest dostępna na następujących platformach:

- AIX
- Linux
- Windows
- z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Nazwa formatu

MQFMT_CICS

Wersja

Bieżąca wersja MQCIH to MQCIH_VERSION_2. Pola, które istnieją tylko w nowszej wersji struktury, są identyfikowane jako takie w kolejnych opisach.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję produktu MQCIH z wartością początkową pola *Version* ustawioną na MQCIH_VERSION_2.

Zestaw znaków i kodowanie

Warunki specjalne mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQCIH i danych komunikatu aplikacji:

- Aplikacje nawiązujące połączenie z menedżerem kolejek, do którego należy kolejka CICS bridge , muszą udostępniać strukturę MQCIH, która jest w zestawie znaków i kodowaniu menedżera kolejek. Jest to spowodowane tym, że w tym przypadku nie jest wykonywana konwersja danych struktury MQCIH.
- Aplikacje łączące się z innymi menedżerami kolejek mogą udostępniać strukturę MQCIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań. Agent odbierającego kanału komunikatów nawiązał połączenie z menedżerem kolejek, który jest właścicielem kolejki CICS bridge , przekształca strukturę MQCIH.
- Dane komunikatu aplikacji po strukturze MQCIH muszą być w tym samym zestawie znaków i kodowaniu, co struktura MQCIH. Pól *CodedCharSetId* i *Encoding* w strukturze MQCIH nie można używać do określania zestawu znaków i kodowania danych komunikatu aplikacji.

  Należy udostępnić wyjście konwersji danych w celu przekształcenia danych komunikatu aplikacji, jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek.

Użycie

Jeśli aplikacja wymaga wartości, które są takie same jak wartości początkowe przedstawione w sekcji Tabela 477 na stronie 305, a most działa z opcją AUTH=LOCAL lub AUTH=IDENTIFY, można pominąć strukturę MQCIH w komunikacie. We wszystkich innych przypadkach struktura musi być obecna.

Most akceptuje strukturę MQCIH version-1 lub version-2 , ale w przypadku transakcji 3270 należy użyć struktury version-2 .

Aplikacja musi zapewnić, że pola udokumentowane jako pola żądania mają odpowiednie wartości w komunikacie wysyłanym do mostu. Te pola są polami wejściowymi do mostu.

Pola udokumentowane jako pola odpowiedzi są ustawiane przez CICS bridge w komunikacie odpowiedzi wysyłanym przez most do aplikacji. Informacje o błędach są zwracane w polach *ReturnCode*, *Function*, *CompCode*, *Reason*i *AbendCode* , ale nie wszystkie z nich są ustawiane we wszystkich przypadkach. W poniższej tabeli przedstawiono, które pola są ustawione dla różnych wartości *ReturnCode*.

Tabela 476. Zawartość pól informacji o błędach w strukturze MQCIH dla MQCIH

ReturnCode	***Function***	***CompCode***	***Reason***	***AbendCode***
MQCRC_OK	-	-	-	-
BŁĄD MQCRC_BRIDGE_ERROR	-	-	MQFB_CICS_ *	-
MQCRC_MQ_API_ERROR MQCRC_BRIDGE_TIMEOUT	MQ call name (nazwa połączenia)	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	-	-	-	CICS KOD ABCODE

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 477. Pola w MQCIH dla MQCIH

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQCIH_ID_struktury	'CIH¬'
Wersja (numer wersji struktury)	MQCIH_VERSION_2	2
StrucLength (długość struktury MQCIH)	MQCIH_LENGTH_2	180
Kodowanie (zarezerwowane)	Brak	0
CodedCharSetId (zarezerwowany)	Brak	0
Format (nazwa formatu produktuMQ dla danych następujących po MQCIH)	MQFMT_BRAK	Puste
Flagi (flags)	MQCIH_BRAK	0
ReturnCode (kod powrotu z mostu)	MQCRC_OK	0
CompCode (kod zakończeniaMQ lub CICS EIBRESP)	MQCC_OK	0
Przyczyna (kod przyczyny lub opinii produktuMQ lub CICS EIBRESP2)	MQRC_BRAK	0
UOWControl (kontrola jednostki pracy)	TYLKO MQCUOWC_ONLY	273
GetWaitInterwał (czas oczekiwania na wywołanie MQGET wydane przez zadanie mostu)	MQCGWI_DEFAULT	-2
LinkType (typ powiązania)	MQCLT_PROGRAM	1
OutputDataDługość (długość danych wyjściowych COMMAREA)	MQCODL_AS_INPUT	-1
FacilityKeepCzas (czas zwolnienia narzędzia mostu)	Brak	0
ADSDescriptor (deskryptor ADS)	MQCADSD_BRAK	0
ConversationalTask (określa, czy zadanie może być konwersacyjne)	MQCCT_NO	0
TaskEndStatus (status na końcu zadania)	MQCTES_NOSYNC	0
Facility (token narzędzia mostu)	MQCFAC_BRAK	Wartości null
Funkcja (nazwa wywołaniaMQ lub funkcja CICS EIBFN)	MQCFUNC_BRAK	Puste
AbendCode (kod nieprawidłowego zakończenia)	Brak	Puste
Element uwierzytelniający (hasło lub przepustka)	Brak	Puste
Reserved1 (zarezerwowany)	Brak	Puste
ReplyToFormat (nazwa komunikatu odpowiedzi w formacieMQ)	MQFMT_BRAK	Puste

Tabela 477. Pola w MQCIH dla MQCIH (kontynuacja)

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
RemoteSysID (identyfikator zdalnego systemu CICS , który ma być używany)	Brak	Puste
RemoteTransID (CICS RTRANSID do użycia)	Brak	Puste
TransactionId (transakcja do przyłączenia)	Brak	Puste
FacilityLike (emulowane atrybuty terminalu)	Brak	Puste
AttentionId (klawisz AID)	Brak	Puste
StartCode (kod początkowy transakcji)	MQCSC_BRAK	Puste
CancelCode (kod nieprawidłowego zakończenia transakcji)	Brak	Puste
NextTransactionId (następna transakcja do przyłączenia)	Brak	Puste
Reserved2 (zarezerwowany)	Brak	Puste
Reserved3 (zarezerwowany)	Brak	Puste
Uwaga: Pozostałe pola nie są wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.		
CursorPosition (pozycja kursora)	Brak	0
ErrorOffset (przesunięcie błędu w komunikacie)	Brak	0
InputItem (element wejściowy)	Brak	0
Reserved4 (zarezerwowany)	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQCIH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCIH MyCIH = {MQCIH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQCIH

```
typedef struct tagMQCIH MQCIH;
struct tagMQCIH {
  MQCHAR4  StrucId;             /* Structure identifier */
  MQLONG   Version;             /* Structure version number */
  MQLONG   StrucLength;         /* Length of MQCIH structure */
  MQLONG   Encoding;            /* Reserved */
  MQLONG   CodedCharSetId;      /* Reserved */
  MQCHAR8  Format;              /* MQ format name of data that follows
                                   MQCIH */
  MQLONG   Flags;               /* Flags */
  MQLONG   ReturnCode;          /* Return code from bridge */
  MQLONG   CompCode;            /* MQ completion code or CICS EIBRESP */
  MQLONG   Reason;              /* MQ reason or feedback code, or CICS
                                   EIBRESP2 */
  MQLONG   UOWControl;          /* Unit-of-work control */
```

```
  MQLONG   GetWaitInterval;     /* Wait interval for MQGET call issued
                                   by bridge task */
  MQLONG   LinkType;            /* Link type */
  MQLONG   OutputDataLength;    /* Output COMMAREA data length */
  MQLONG   FacilityKeepTime;    /* Bridge facility release time */
  MQLONG   ADSDescriptor;       /* Send/receive ADS descriptor */
  MQLONG   ConversationalTask;  /* Whether task can be conversational */
  MQLONG   TaskEndStatus;       /* Status at end of task */
  MQBYTE8  Facility;            /* Bridge facility token */
  MQCHAR4  Function;            /* MQ call name or CICS EIBFN
                                   function */
  MQCHAR4  AbendCode;           /* Abend code */
  MQCHAR8  Authenticator;       /* Password or passticket */
  MQCHAR8  Reserved1;           /* Reserved */
  MQCHAR8  ReplyToFormat;       /* MQ format name of reply message */
  MQCHAR4  RemoteSysId;         /* Reserved */
  MQCHAR4  RemoteTransId;       /* Reserved */
  MQCHAR4  TransactionId;       /* Transaction to attach */
  MQCHAR4  FacilityLike;        /* Terminal emulated attributes */
  MQCHAR4  AttentionId;         /* AID key */
  MQCHAR4  StartCode;           /* Transaction start code */
  MQCHAR4  CancelCode;          /* Abend transaction code */
  MQCHAR4  NextTransactionId;   /* Next transaction to attach */
  MQCHAR8  Reserved2;           /* Reserved */
  MQCHAR8  Reserved3;           /* Reserved */
  MQLONG   CursorPosition;      /* Cursor position */
  MQLONG   ErrorOffset;         /* Offset of error in message */
  MQLONG   InputItem;           /* Reserved */
  MQLONG   Reserved4;           /* Reserved */
};
```

Deklaracja języka COBOL dla MQCIH

```
**   MQCIH structure
  10 MQCIH.
**    Structure identifier
   15 MQCIH-STRUCID            PIC X(4).
**    Structure version number
   15 MQCIH-VERSION            PIC S9(9) BINARY.
**    Length of MQCIH structure
   15 MQCIH-STRUCLENGTH        PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-ENCODING           PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-CODEDCHARSETID     PIC S9(9) BINARY.
**    MQ format name of data that follows MQCIH
   15 MQCIH-FORMAT             PIC X(8).
**    Flags
   15 MQCIH-FLAGS              PIC S9(9) BINARY.
**    Return code from bridge
   15 MQCIH-RETURNCODE         PIC S9(9) BINARY.
**    MQ completion code or CICS EIBRESP
   15 MQCIH-COMPCODE           PIC S9(9) BINARY.
**    MQ reason or feedback code, or CICS EIBRESP2
   15 MQCIH-REASON             PIC S9(9) BINARY.
**    Unit-of-work control
   15 MQCIH-UOWCONTROL         PIC S9(9) BINARY.
**    Wait interval for MQGET call issued by bridge task
   15 MQCIH-GETWAITINTERVAL    PIC S9(9) BINARY.
**    Link type
   15 MQCIH-LINKTYPE           PIC S9(9) BINARY.
**    Output COMMAREA data length
   15 MQCIH-OUTPUTDATALENGTH   PIC S9(9) BINARY.
**    Bridge facility release time
   15 MQCIH-FACILITYKEEPTIME   PIC S9(9) BINARY.
**    Send/receive ADS descriptor
   15 MQCIH-ADSDESCRIPTOR      PIC S9(9) BINARY.
**    Whether task can be conversational
   15 MQCIH-CONVERSATIONALTASK PIC S9(9) BINARY.
**    Status at end of task
   15 MQCIH-TASKENDSTATUS      PIC S9(9) BINARY.
**    Bridge facility token
   15 MQCIH-FACILITY           PIC X(8).
**    MQ call name or CICS EIBFN function
   15 MQCIH-FUNCTION           PIC X(4).
**    Abend code
   15 MQCIH-ABENDCODE          PIC X(4).
**    Password or passticket
   15 MQCIH-AUTHENTICATOR      PIC X(8).
```

```
**    Reserved
   15 MQCIH-RESERVED1          PIC X(8).
**    MQ format name of reply message
   15 MQCIH-REPLYTOFORMAT      PIC X(8).
**    Reserved
   15 MQCIH-REMOTESYSID        PIC X(4).
**    Reserved
   15 MQCIH-REMOTETRANSID      PIC X(4).
**    Transaction to attach
   15 MQCIH-TRANSACTIONID      PIC X(4).
**    Terminal emulated attributes
   15 MQCIH-FACILITYLIKE       PIC X(4).
**    AID key
   15 MQCIH-ATTENTIONID        PIC X(4).
**    Transaction start code
   15 MQCIH-STARTCODE          PIC X(4).
**    Abend transaction code
   15 MQCIH-CANCELCODE         PIC X(4).
**    Next transaction to attach
   15 MQCIH-NEXTTRANSACTIONID  PIC X(4).
**    Reserved
   15 MQCIH-RESERVED2          PIC X(8).
**    Reserved
   15 MQCIH-RESERVED3          PIC X(8).
**    Cursor position
   15 MQCIH-CURSORPOSITION     PIC S9(9) BINARY.
**    Offset of error in message
   15 MQCIH-ERROROFFSET        PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-INPUTITEM          PIC S9(9) BINARY.
**    Reserved
   15 MQCIH-RESERVED4          PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQCIH

```
dcl
 1 MQCIH based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 StrucLength        fixed bin(31), /* Length of MQCIH structure */
  3 Encoding           fixed bin(31), /* Reserved */
  3 CodedCharSetId     fixed bin(31), /* Reserved */
  3 Format             char(8),       /* MQ format name of data that
                                         follows MQCIH */
  3 Flags              fixed bin(31), /* Flags */
  3 ReturnCode         fixed bin(31), /* Return code from bridge */
  3 CompCode           fixed bin(31), /* MQ completion code or CICS
                                         EIBRESP */
  3 Reason             fixed bin(31), /* MQ reason or feedback code, or
                                         CICS EIBRESP2 */
  3 UOWControl         fixed bin(31), /* Unit-of-work control */
  3 GetWaitInterval    fixed bin(31), /* Wait interval for MQGET call
                                         issued by bridge task */
  3 LinkType           fixed bin(31), /* Link type */
  3 OutputDataLength   fixed bin(31), /* Output COMMAREA data length */
  3 FacilityKeepTime   fixed bin(31), /* Bridge facility release time */
  3 ADSDescriptor      fixed bin(31), /* Send/receive ADS descriptor */
  3 ConversationalTask fixed bin(31), /* Whether task can be
                                         conversational */
  3 TaskEndStatus      fixed bin(31), /* Status at end of task */
  3 Facility           char(8),       /* Bridge facility token */
  3 Function           char(4),       /* MQ call name or CICS EIBFN
                                         function */
  3 AbendCode          char(4),       /* Abend code */
  3 Authenticator      char(8),       /* Password or passticket */
  3 Reserved1          char(8),       /* Reserved */
  3 ReplyToFormat      char(8),       /* MQ format name of reply
                                         message */
  3 RemoteSysId        char(4),       /* Reserved */
  3 RemoteTransId      char(4),       /* Reserved */
  3 TransactionId      char(4),       /* Transaction to attach */
  3 FacilityLike       char(4),       /* Terminal emulated attributes */
  3 AttentionId        char(4),       /* AID key */
  3 StartCode          char(4),       /* Transaction start code */
  3 CancelCode         char(4),       /* Abend transaction code */
  3 NextTransactionId  char(4),       /* Next transaction to attach */
  3 Reserved2          char(8),       /* Reserved */
  3 Reserved3          char(8),       /* Reserved */
  3 CursorPosition     fixed bin(31), /* Cursor position */
```

```
  3 ErrorOffset        fixed bin(31), /* Offset of error in message */
  3 InputItem          fixed bin(31), /* Reserved */
  3 Reserved4          fixed bin(31); /* Reserved */
```

Deklaracja High Level Assembler dla MQCIH

```
MQCIH                     DSECT
MQCIH_STRUCID             DS   CL4  Structure identifier
MQCIH_VERSION             DS   F    Structure version number
MQCIH_STRUCLENGTH         DS   F    Length of MQCIH structure
MQCIH_ENCODING            DS   F    Reserved
MQCIH_CODEDCHARSETID      DS   F    Reserved
MQCIH_FORMAT              DS   CL8  MQ format name of data that follows
*                                   MQCIH
MQCIH_FLAGS               DS   F    Flags
MQCIH_RETURNCODE          DS   F    Return code from bridge
MQCIH_COMPCODE            DS   F    MQ completion code or CICS EIBRESP
MQCIH_REASON              DS   F    MQ reason or feedback code, or CICS
*                                   EIBRESP2
MQCIH_UOWCONTROL          DS   F    Unit-of-work control
MQCIH_GETWAITINTERVAL     DS   F    Wait interval for MQGET call issued
*                                   by bridge task
MQCIH_LINKTYPE            DS   F    Link type
MQCIH_OUTPUTDATALENGTH    DS   F    Output COMMAREA data length
MQCIH_FACILITYKEEPTIME    DS   F    Bridge facility release time
MQCIH_ADSDESCRIPTOR       DS   F    Send/receive ADS descriptor
MQCIH_CONVERSATIONALTASK  DS   F    Whether task can be conversational
MQCIH_TASKENDSTATUS       DS   F    Status at end of task
MQCIH_FACILITY            DS   XL8  Bridge facility token
MQCIH_FUNCTION            DS   CL4  MQ call name or CICS EIBFN function
MQCIH_ABENDCODE           DS   CL4  Abend code
MQCIH_AUTHENTICATOR       DS   CL8  Password or passticket
MQCIH_RESERVED1           DS   CL8  Reserved
MQCIH_REPLYTOFORMAT       DS   CL8  MQ format name of reply message
MQCIH_REMOTESYSID         DS   CL4  Reserved
MQCIH_REMOTETRANSID       DS   CL4  Reserved
MQCIH_TRANSACTIONID       DS   CL4  Transaction to attach
MQCIH_FACILITYLIKE        DS   CL4  Terminal emulated attributes
MQCIH_ATTENTIONID         DS   CL4  AID key
MQCIH_STARTCODE           DS   CL4  Transaction start code
MQCIH_CANCELCODE          DS   CL4  Abend transaction code
MQCIH_NEXTTRANSACTIONID   DS   CL4  Next transaction to attach
MQCIH_RESERVED2           DS   CL8  Reserved
MQCIH_RESERVED3           DS   CL8  Reserved
MQCIH_CURSORPOSITION      DS   F    Cursor position
MQCIH_ERROROFFSET         DS   F    Offset of error in message
MQCIH_INPUTITEM           DS   F    Reserved
MQCIH_RESERVED4           DS   F    Reserved
*
MQCIH_LENGTH              EQU  *-MQCIH
                          ORG  MQCIH
MQCIH_AREA                DS   CL(MQCIH_LENGTH)
```

Deklaracja Visual Basic dla MQCIH

```
Type MQCIH
  StrucId            As String*4 'Structure identifier'
  Version            As Long     'Structure version number'
  StrucLength        As Long     'Length of MQCIH structure'
  Encoding           As Long     'Reserved'
  CodedCharSetId     As Long     'Reserved'
  Format             As String*8 'MQ format name of data that follows'
                                 'MQCIH'
  Flags              As Long     'Flags'
  ReturnCode         As Long     'Return code from bridge'
  CompCode           As Long     'MQ completion code or CICS EIBRESP'
  Reason             As Long     'MQ reason or feedback code, or CICS'
                                 'EIBRESP2'
  UOWControl         As Long     'Unit-of-work control'
  GetWaitInterval    As Long     'Wait interval for MQGET call issued'
                                 'by bridge task'
  LinkType           As Long     'Link type'
  OutputDataLength   As Long     'Output COMMAREA data length'
  FacilityKeepTime   As Long     'Bridge facility release time'
  ADSDescriptor      As Long     'Send/receive ADS descriptor'
  ConversationalTask As Long     'Whether task can be conversational'
  TaskEndStatus      As Long     'Status at end of task'
```

```
   Facility           As MQBYTE8  'Bridge facility token'
   Function           As String*4 'MQ call name or CICS EIBFN function'
   AbendCode          As String*4 'Abend code'
   Authenticator      As String*8 'Password or passticket'
   Reserved1          As String*8 'Reserved'
   ReplyToFormat      As String*8 'MQ format name of reply message'
   RemoteSysId        As String*4 'Reserved'
   RemoteTransId      As String*4 'Reserved'
   TransactionId      As String*4 'Transaction to attach'
   FacilityLike       As String*4 'Terminal emulated attributes'
   AttentionId        As String*4 'AID key'
   StartCode          As String*4 'Transaction start code'
   CancelCode         As String*4 'Abend transaction code'
   NextTransactionId  As String*4 'Next transaction to attach'
   Reserved2          As String*8 'Reserved'
   Reserved3          As String*8 'Reserved'
   CursorPosition     As Long     'Cursor position'
   ErrorOffset        As Long     'Offset of error in message'
   InputItem          As Long     'Reserved'
   Reserved4          As Long     'Reserved'
End Type
```

StrucId (MQCHAR4) dla MQCIH

Jest to identyfikator struktury struktury nagłówka informacji CICS . Jest to zawsze pole wejściowe. Jego wartością jest MQCIH_STRUC_ID.

Wartość musi być następująca:

MQCIH_ID_struktury
Identyfikator struktury nagłówka informacji CICS .

Dla języka programowania C zdefiniowana jest również stała MQCIH_STRUC_ID_ARRAY. Ma taką samą wartość jak MQCIH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQCIH

To pole jest polem żądania. Wartością początkową jest MQCIH_VERSION_2.

Wartość musi być jedną z następujących wartości:

MQCIH_VERSION_1
Version-1 CICS struktura nagłówka informacji.

MQCIH_VERSION_2
Version-2 CICS struktura nagłówka informacji.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQCIH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka informacji CICS .

StrucLength (MQLONG) dla MQCIH

To pole jest polem żądania o wartości początkowej MQCIH_LENGTH_2.

Wartość musi być jedną z następujących wartości:

MQCIH_LENGTH_1
Długość struktury nagłówka informacji version-1 CICS .

MQCIH_LENGTH_2
Długość struktury nagłówka informacji version-2 CICS .

Następująca stała określa długość bieżącej wersji:

MQCIH_CURRENT_LENGTH
Długość bieżącej wersji struktury nagłówka informacji CICS .

Kodowanie (MQLONG) dla MQCIH

To pole jest polem zarezerwowanym; jego wartość nie jest istotna. Wartością początkową jest 0.

Kodowanie dla obsługiwanych struktur, które są zgodne ze strukturą MQCIH, jest takie samo jak kodowanie samej struktury MQCIH i jest pobierane z dowolnego poprzedzającego nagłówka IBM MQ .

CodedCharSetId (MQLONG) dla MQCIH

CodedCharSetId jest polem zarezerwowanym, a jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

Identyfikator zestawu znaków dla obsługiwanych struktur, które są zgodne ze strukturą MQCIH, jest taki sam jak identyfikator zestawu znaków samej struktury MQCIH i jest pobierany z dowolnego poprzedzającego nagłówka IBM MQ .

Format (MQCHAR8) dla MQCIH

W tym polu jest wyświetlana nazwa formatu IBM MQ danych, które są zgodne ze strukturą MQCIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w strukturze MQMD.

Ta nazwa formatu jest również używana dla komunikatu odpowiedzi, jeśli pole *ReplyToFormat* ma wartość MQFMT_NONE.

- W przypadku żądań DPL *Format* musi być nazwą formatu obszaru COMMAREA.
- W przypadku żądań 3270 *Format* musi mieć wartość CSQCBDCI, a most ustawia format CSQCBDC0 dla komunikatów odpowiedzi.

Wyjścia konwersji danych dla tych formatów muszą być zainstalowane w menedżerze kolejek, w którym mają być uruchamiane.

Jeśli komunikat żądania generuje komunikat odpowiedzi o błędzie, komunikat odpowiedzi o błędzie ma nazwę formatu MQFMT_STRING.

To pole jest polem żądania. Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQCIH

To pole jest polem żądania. Wartością początkową tego pola jest MQCIH_NONE.

Wartość musi być następująca:

MQCIH_BRAK
Brak flag.

MQCIH_PASS_EXPIRATION
Komunikat odpowiedzi zawiera:

- Te same opcje raportu utraty ważności, co w przypadku komunikatu żądania.
- Pozostały czas utraty ważności z komunikatu żądania bez korekty czasu przetwarzania mostu.

Jeśli ta wartość zostanie pominięta, czas utraty ważności zostanie ustawiony na *nieograniczony*.

MQCIH_REPLY_WITHOUT_NULLS
Długość komunikatu odpowiedzi dla żądania programu CICS DPL jest dopasowywana w celu wykluczenia końcowych wartości NULL (X'00 ') na końcu obszaru COMMAREA zwróconego przez program DPL. Jeśli ta wartość nie jest ustawiona, wartości null mogą być istotne i zwracany jest pełny obszar COMMAREA.

MQCIH_SYNC_ON_RETURN
Łącze CICS dla żądań DPL korzysta z opcji SYNCONRETURN, co powoduje, że CICS będzie przyjmowało punkt synchronizacji po zakończeniu działania programu, jeśli zostanie on dostarczony do innego regionu CICS . Most nie określa, do którego regionu CICS ma zostać wysłane żądanie, który jest kontrolowany przez definicję programu CICS lub narzędzia równoważenia obciążenia.

ReturnCode (MQLONG) dla MQCIH
Wartością tego pola jest kod powrotu z CICS bridge opisujący wynik przetwarzania wykonywanego przez most. To pole jest polem odpowiedzi z wartością początkową MQCRC_OK.

Pola *Function*, *CompCode*, *Reason*i *AbendCode* mogą zawierać dodatkowe informacje (patrz sekcja Tabela 476 na stronie 304). Jest to jedna z następujących wartości:

MQCRC_APPLICATION_ABEND (niepoprawne zakończenie aplikacji MQ)
(5, X'005 ') Aplikacja została zakończona nieprawidłowo.

MQCRC_BRIDGE_ABEND, nieprawidłowe zakończenie
(4, X'004 ') CICS bridge zakończyło się nieprawidłowo.

BŁĄD MQCRC_BRIDGE_ERROR
(3, X'003 ') CICS bridge wykrył błąd.

MQCRC_BRIDGE_TIMEOUT,
(8, X'008 ') Drugi lub późniejszy komunikat w bieżącej jednostce pracy nie został odebrany w określonym czasie.

BŁĄD WYKONYWANIA MQCRC_CICS_EXEC_ERROR
(1, X'001 ') Instrukcja EXEC CICS wykryła błąd.

BŁĄD MQCRC_MQ_API_ERROR
(2, X'002 ') Wywołanie produktu MQ wykryło błąd.

MQCRC_OK
(0, X'000 ') Brak błędu.

MQCRC_PROGRAM_NIEDOSTĘPNE
(7, X'007 ') Program niedostępny.

BŁĄD MQCRC_SECURITY_ERROR
(6, X'006 ') Wystąpił błąd ochrony.

MQCRC_TRANSID_NIEDOSTĘPNE
(9, X'009 ') Transakcja niedostępna.

CompCode (MQLONG) dla MQCIH
To pole jest polem odpowiedzi. Jego wartością początkową jest MQCC_OK

Wartość zwracana w tym polu zależy od wartości *ReturnCode* ; zawiera sekcja Tabela 476 na stronie 304.

Przyczyna (MQLONG) MQCIH
To pole jest polem odpowiedzi. Jego wartością początkową jest MQRC_NONE.

Wartość zwracana w tym polu zależy od wartości *ReturnCode* ; zawiera sekcja Tabela 476 na stronie 304.

UOWControl (MQLONG) dla MQCIH
To pole jest polem żądania, które steruje przetwarzaniem jednostki pracy wykonywanym przez CICS bridge. Wartością początkową tego pola jest MQCUOWC_ONLY.

Można zażądać od mostu uruchomienia pojedynczej transakcji lub jednego lub większej liczby programów w ramach jednostki pracy. Pole wskazuje, czy CICS bridge uruchamia jednostkę pracy, wykonuje żądaną funkcję w bieżącej jednostce pracy, czy kończy jednostkę pracy, zatwierdzając ją lub cofając. W celu zoptymalizowania przepływów transmisji danych obsługiwane są różne kombinacje.

Wartość musi być jedną z następujących wartości:

TYLKO MQCUOWC_ONLY
Uruchom jednostkę pracy, wykonaj funkcję, a następnie zatwierdź jednostkę pracy.

MQCUOWC_CONTINUE
Dodatkowe dane dla bieżącej jednostki pracy (tylko 3270).

MQCUOWC_FIRST
Uruchom jednostkę pracy i wykonaj funkcję.

MQCUOWC_MIDDLE,
Wykonaj funkcję w bieżącej jednostce pracy

MQCUOWC_LAST
Wykonaj funkcję, a następnie zatwierdź jednostkę pracy.

MQCUOWC_COMMIT
Zatwierdź jednostkę pracy (tylko DPL).

MQCUOWC_BACKOUT,
Wycofuje jednostkę pracy (tylko DPL).

GetWaitInterwał (MQLONG) dla MQCIH
To pole jest polem żądania. Jego wartością początkową jest MQCGWI_DEFAULT.

To pole ma zastosowanie tylko wtedy, gdy *UOWControl* ma wartość MQCUOWC_FIRST. Umożliwia ona aplikacji wysyłającej określenie przybliżonego czasu (w milisekundach), przez który wywołania MQGET wysyłane przez most będą oczekiwać na drugi i kolejny komunikat żądania dla jednostki pracy uruchomionej przez ten komunikat. Ta funkcja nadpisuje domyślny odstęp czasu oczekiwania używany przez most. Można użyć następujących wartości specjalnych:

MQCGWI_DEFAULT
Domyślny odstęp czasu oczekiwania.

Ta wartość powoduje, że CICS bridge oczekuje przez czas określony podczas uruchamiania mostu.

MQWI_UNLIMITED
Nieograniczony czas oczekiwania.

LinkType (MQLONG) dla MQCIH
To pole jest polem żądania. Jego wartością początkową jest MQCLT_PROGRAM.

Ta wartość wskazuje typ obiektu, który most próbuje połączyć. Musi to być jedna z następujących wartości:

MQCLT_PROGRAM
Program DPL.

MQCLT_TRANSACTION
Transakcja 3270.

OutputDataDługość (MQLONG) dla MQCIH
To pole jest polem żądania używanym tylko dla programów DPL. Jego wartością początkową jest MQCODL_AS_INPUT.

Ta wartość jest długością danych użytkownika, które mają zostać zwrócone klientowi w komunikacie odpowiedzi. Ta długość obejmuje 8-bajtową nazwę programu. Długość obszaru COMMAREA przekazanego do programu dowiązanego jest maksymalną wartością tego pola i maksymalną długością danych użytkownika w komunikacie żądania pomniejszoną o 8.

Uwaga: Długość danych użytkownika w komunikacie to długość komunikatu bez struktury MQCIH.

Jeśli długość danych użytkownika w komunikacie żądania jest mniejsza niż *OutputDataLength*, używana jest opcja DATALENGTH komendy LINK , co pozwala na efektywne przesłanie funkcji LINK do innego regionu CICS .

Można użyć następującej wartości specjalnej:

MQCODL_AS_INPUT
Długość danych wyjściowych jest taka sama jak długość danych wejściowych.

Ta wartość może być potrzebna nawet wtedy, gdy nie jest wymagana żadna odpowiedź, aby zapewnić, że obszar COMMAREA przekazany do programu połączonego ma wystarczającą wielkość.

FacilityKeepCzas (MQLONG) dla MQCIH

FacilityKeepCzas (w sekundach) przechowywania narzędzia mostu po zakończeniu transakcji użytkownika.

W przypadku transakcji pseudo-konwersacyjnych należy podać wartość odpowiadającą oczekiwanemu czasowi trwania pseudo-konwersacji; należy podać zero dla ostatniej transakcji pseudo-konwersacji, a dla innych typów transakcji-zero.

To pole jest polem żądania używanym tylko dla transakcji 3270. Wartością początkową tego pola jest 0.

ADSDescriptor (MQLONG) dla MQCIH

To pole jest indykatorem określającym, czy deskryptory ADS mają być wysyłane w żądaniach SEND i RECEIVE BMS.

Zdefiniowane są następujące wartości:

MQCADSD_BRAK
Nie wysyłaj ani nie odbieraj deskryptorów ADS.

MQCADSD_SEND
Wyślij deskryptory ADS.

MQCADSD_RECV
Odbierz deskryptory ADS.

MQCADSD_MSGFORMAT
Użyj formatu komunikatu dla deskryptorów ADS.

Powoduje to wysłanie lub odebranie deskryptorów ADS przy użyciu długiej postaci deskryptora ADS. Długi formularz zawiera pola, które są wyrównane do 4-bajtowych granic.

Ustaw pole *ADSDescriptor* w następujący sposób:

- Jeśli deskryptory ADS nie są używane, należy ustawić w tym polu wartość MQCADSD_NONE.
- Jeśli w każdym środowisku używane są deskryptory ADS z *takim samym* identyfikatorem CCSID, ustaw pole na sumę wartości MQCADSD_SEND i MQCADSD_RECV.
- Jeśli używane są deskryptory ADS z *różnymi* identyfikatorami CCSID w każdym środowisku, ustaw pole na sumę wartości MQCADSD_SEND, MQCADSD_RECV i MQCADSD_MSGFORMAT.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest MQCADSD_NONE.

ConversationalTask (MQLONG) dla MQCIH

To pole jest indykatorem określającym, czy zadanie może wysyłać żądania dotyczące dodatkowych informacji, czy też zatrzymywać zadanie i wysyłać komunikat o awariach.

Wartość musi być jedną z następujących opcji:

MQCCT_YES
Zadanie jest konwersacyjne.

MQCCT_NO
Zadanie nie jest konwersacyjne.

To pole jest polem żądania używanym tylko dla transakcji 3270. Wartością początkową tego pola jest MQCCT_NO.

TaskEndStatus (MQLONG) dla MQCIH

To pole jest polem odpowiedzi przedstawiającym status transakcji użytkownika na końcu zadania. To pole jest używane tylko dla transakcji 3270, a jego wartością początkową jest MQCTES_NOSYNC.

Zwracana jest jedna z następujących wartości:

MQCTES_NOSYNC
Niezsynchronizowane.

Transakcja użytkownika nie została jeszcze zakończona i nie została zsynchronizowana. W tym przypadku pole *MsgType* w strukturze MQMD ma wartość MQMT_REQUEST.

MQCTES_COMMIT
Zatwierdź jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona, ale została zsynchronizowana z pierwszą jednostką pracy. W tym przypadku pole *MsgType* w strukturze MQMD to MQMT_DATAGRAM.

MQCTES_BACKOUT
Wycofuje jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona. Bieżąca jednostka pracy została wycofana. W tym przypadku pole *MsgType* w strukturze MQMD to MQMT_DATAGRAM.

MQCTES_ENDTASK
Zakończ zadanie.

Transakcja użytkownika została zakończona (lub zakończona awaryjnie). W tym przypadku pole *MsgType* w strukturze MQMD ma wartość MQMT_REPLY.

Narzędzie (MQBYTE8) dla MQCIH

W tym polu wyświetlany jest 8-bajtowy token narzędzia mostu.

Token narzędzia mostu umożliwia wielu transakcjom w pseudokonwersacji korzystanie z tego samego narzędzia mostu (wirtualnego terminalu 3270). W pierwszym lub jedynym komunikacie w pseudokonwersacji ustaw wartość MQCFAC_NONE. Ta wartość informuje produkt CICS o konieczności przydzielenia nowego narzędzia mostu dla tego komunikatu. Znacznik narzędzia mostu jest zwracany w komunikatach odpowiedzi, jeśli w komunikacie wejściowym określono niezerową wartość *FacilityKeepTime* . Kolejne komunikaty wejściowe w pseudokonwersacji muszą następnie używać tego samego znacznika narzędzia mostu.

Zdefiniowana jest następująca wartość specjalna:

MQCFAC_BRAK
Nie określono tokenu narzędzia.

W języku programowania C stała MQCFAC_NONE_ARRAY jest również zdefiniowana i ma taką samą wartość jak MQCFAC_NONE, ale jest tablicą znaków zamiast łańcucha.

To pole jest zarówno polem żądania, jak i polem odpowiedzi używanym tylko dla transakcji 3270. Długość tego pola jest określona przez parametr MQ_FACILITY_LENGTH. Wartością początkową tego pola jest MQCFAC_NONE.

Funkcja (MQCHAR4) dla MQCIH

To pole jest polem odpowiedzi. Długość tego pola jest określona przez parametr MQ_FUNCTION_LENGTH. Wartością początkową tego pola jest MQCFUNC_NONE.

Wartość zwracana w tym polu zależy od wartości *ReturnCode* ; zawiera sekcja Tabela 476 na stronie 304. Jeśli plik *Function* zawiera nazwę wywołania IBM MQ , możliwe są następujące wartości:

MQCFUNC_MQCONN,
Wywołanie MQCONN.

MQCFUNC_MQGET,
Wywołanie MQGET.

MQCFUNC_MQINQ,
Wywołanie MQINQ.

MQCFUNC_MQOPEN,
Wywołanie MQOPEN.

MQCFUNC_MQPUT,
Wywołanie MQPUT.

MQCFUNC_MQPUT1
Wywołanie MQPUT1 .

MQCFUNC_BRAK
Brak połączenia.

We wszystkich przypadkach dla języka programowania C zdefiniowane są również stałe MQCFUNC_ * _ARRAY. Te stałe mają takie same wartości jak odpowiadające im stałe MQCFUNC_ *, ale są tablicami znaków, a nie łańcuchami.

AbendCode (MQCHAR4) dla MQCIH
AbendCode to pole odpowiedzi. Długość tego pola jest określona przez wartość MQ_ABEND_CODE_LENGTH. Wartością początkową tego pola są 4 znaki odstępu.

Wartość zwracana w tym polu jest istotna tylko wtedy, gdy pole *ReturnCode* ma wartość MQCRC_APPLICATION_ABEND lub MQCRC_BRIDGE_ABEND. Jeśli tak, *AbendCode* zawiera wartość CICS ABCODE.

Element uwierzytelniający (MQCHAR8) dla MQCIH
Wartością tego pola jest hasło lub przepustka.

Jeśli uwierzytelnianie za pomocą identyfikatora użytkownika jest aktywne dla CICS bridge, do uwierzytelniania nadawcy komunikatu używany jest identyfikator *Authenticator* z identyfikatorem użytkownika w kontekście tożsamości MQMD.

To jest pole żądania. Długość tego pola jest określona przez parametr MQ_AUTHENTICATOR_LENGTH. Wartością początkową tego pola jest 8 odstępów.

Reserved1 (MQCHAR8) dla MQCIH
To pole jest polem zarezerwowanym. Wartość musi być równa 8 odstępów.

Format ReplyTo(MQCHAR8) dla MQCIH
Wartością tego pola jest nazwa formatu IBM MQ komunikatu odpowiedzi, który jest wysyłany w odpowiedzi na bieżący komunikat.

Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w MQMD.

To pole jest polem żądania używanym tylko dla programów DPL. Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

RemoteSysID (MQCHAR4) dla MQCIH
W tym polu wyświetlany jest identyfikator systemu CICS systemu CICS przetwarzającego żądanie.

Jeśli to pole jest puste, żądanie systemowe CICS jest przetwarzane w tym samym systemie CICS , co monitor mostu. Używany identyfikator SYSID jest zwracany w komunikacie odpowiedzi.

W przypadku pseudo-konwersacji 3270 wszystkie kolejne komunikaty w konwersacji muszą określać zdalny identyfikator SYSID zwracany w odpowiedzi początkowej. Jeśli określono, identyfikator SYSID musi:

- Bądź aktywny.
- Mają dostęp do kolejki żądań IBM MQ .
- Dostęp można uzyskać za pośrednictwem łączy ISC CICS z systemu CICS monitora mostu.

RemoteTransID (MQCHAR4) dla MQCIH
To pole jest polem opcjonalnym żądania. Długość tego pola jest określona przez wartość MQ_TRANSACTION_ID_LENGTH.

Jeśli określono, pole jest używane jako wartość RTRANSID parametru CICS START.

TransactionId (MQCHAR4) dla MQCIH

To pole jest polem żądania. Jego długość jest określona przez wartość MQ_TRANSACTION_ID_LENGTH. Wartością początkową tego pola są cztery odstępy.

Jeśli parametr *LinkType* ma wartość MQCLT_TRANSACTION, *TransactionId* jest identyfikatorem transakcji użytkownika, która ma zostać uruchomiona; w tym przypadku należy podać wartość niepustą.

Jeśli parametr *LinkType* ma wartość MQCLT_PROGRAM, *TransactionId* jest kodem transakcji, w ramach którego mają być uruchamiane wszystkie programy w jednostce pracy. Jeśli zostanie podana pusta wartość, zostanie użyty domyślny kod transakcji mostu DPL CICS (CKBP). Jeśli wartość jest niepusta, należy zdefiniować ją w programie CICS jako transakcję lokalną z programem początkowym o wartości CSQCBP00. To pole ma zastosowanie tylko wtedy, gdy *UOWControl* ma wartość MQCUOWC_FIRST lub MQCUOWC_ONLY.

FacilityLike (MQCHAR4) dla MQCIH

FacilityLike to nazwa zainstalowanego terminalu, który ma być używany jako model dla narzędzia mostu.

Wartość pusta oznacza, że wartość *FacilityLike* jest pobierana z definicji profilu transakcji mostu lub używana jest wartość domyślna.

To pole jest polem żądania używanym tylko dla transakcji 3270. Długość tego pola jest określona przez wartość MQ_FACILITY_LIKE_LENGTH. Wartością początkową tego pola są cztery odstępy.

AttentionId (MQCHAR4) dla MQCIH

Wartość w tym polu określa początkową wartość klucza AID podczas uruchamiania transakcji. Jest to wartość jednobajtowa wyrównana do lewej.

Pole AttentionId jest polem żądania używanym tylko dla transakcji 3270. Długość tego pola jest określona przez wartość MQ_ATTENTION_ID_LENGTH. Wartością początkową tego pola są cztery odstępy.

StartCode (MQCHAR4) dla MQCIH

Wartość tego pola jest wskaźnikiem określającym, czy most emuluje transakcję terminalu, czy transakcję zainicjowaną za pomocą komendy START.

Wartość musi być jedną z następujących wartości:

MQCSC_START
Uruchom.

MQCSC_STARTDATA
Dane początkowe.

MQCSC_TERMINPUT
Wejście terminalu.

MQCSC_BRAK
Brak.

We wszystkich przypadkach dla języka programowania C zdefiniowane są również stałe MQCSC_ * _ARRAY. Te stałe mają takie same wartości jak odpowiadające im stałe MQCSC_ *, ale są tablicami znaków, a nie łańcuchami.

W odpowiedzi z mostu to pole jest ustawione na kod początkowy odpowiedni dla następnego identyfikatora transakcji zawartego w polu *NextTransactionId* . W odpowiedzi możliwe są następujące kody początkowe:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT

W przypadku CICS Transaction Server 1.2to pole jest tylko polem żądania; jego wartość w odpowiedzi jest niezdefiniowana.

W przypadku produktu CICS Transaction Server 1.3 i kolejnych wersji to pole jest zarówno polem żądania, jak i polem odpowiedzi.

To pole jest używane tylko dla transakcji 3270. Długość tego pola jest określona przez wartość MQ_START_CODE_LENGTH. Wartością początkową tego pola jest MQCSC_NONE.

CancelCode (MQCHAR4) dla MQCIH
Wartość w tym polu jest kodem nieprawidłowego zakończenia, który ma być używany do zakończenia transakcji (zazwyczaj jest to transakcja konwersacyjna żądająca większej ilości danych). W przeciwnym razie pole to jest puste.

To pole jest polem żądania używanym tylko dla transakcji 3270. Długość tego pola jest określona przez wartość MQ_CANCEL_CODE_LENGTH. Wartością początkową tego pola są cztery odstępy.

Identyfikator NextTransactionId (MQCHAR4) dla MQCIH
Ta wartość jest nazwą następnej transakcji zwróconej przez transakcję użytkownika (zwykle przez EXEC CICS RETURN TRANSID). Jeśli nie ma następnej transakcji, to pole jest puste.

To pole jest polem odpowiedzi używanym tylko dla transakcji 3270. Długość tego pola jest określona przez wartość MQ_TRANSACTION_ID_LENGTH. Wartością początkową tego pola są cztery odstępy.

Reserved2 (MQCHAR8) dla MQCIH
To pole jest polem zarezerwowanym. Wartość musi być równa 8 odstępów.

Reserved3 (MQCHAR8) dla MQCIH
To pole jest polem zarezerwowanym. Wartość musi być równa 8 odstępów.

CursorPosition (MQLONG) (pozycja kursora) dla MQCIH
Wartość w tym polu wskazuje początkową pozycję kursora w momencie rozpoczęcia transakcji.
W przypadku transakcji konwersacyjnych pozycja kursora znajduje się w wektorze RECEIVE.

To pole jest polem żądania używanym tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

ErrorOffset (MQLONG) dla MQCIH
Pole ErrorOffset pokazuje pozycję niepoprawnych danych wykrytych przez wyjście mostu. W tym polu znajduje się przesunięcie od początku komunikatu do miejsca, w którym znajdują się niepoprawne dane.

ErrorOffset jest polem odpowiedzi używanym tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

InputItem (MQLONG) dla MQCIH
To pole jest polem zarezerwowanym. Wartość musi być równa 0.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

Reserved4 (MQLONG) dla MQCIH
To pole jest polem zarezerwowanym. Wartość musi być równa 0.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCIH_VERSION_2.

MQCMHO-opcje tworzenia uchwytu komunikatu

Struktura **MQCMHO** umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia uchwytów komunikatów. Struktura jest parametrem wejściowym wywołania **MQCRTMH** .

Dostępność

Struktura **MQCMHO** jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Zestaw znaków i kodowanie

Dane w pliku **MQCMHO** muszą znajdować się w zestawie znaków aplikacji i kodowania aplikacji (**MQENC_NATIVE**).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 478. Pola w MQCMHO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQCMHO_STRUC_ID (Identyfikator struktury MQCM)	'CMHO'
Wersja (numer wersji struktury)	MQCMHO_VERSION_1	1
Opcje (opcje)	MQCMHO_DEFAULT_VALIDATION,	0

Uwagi:

1. W języku programowania C: zmienna makraParametr MQCMHO_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCMHO MyCMHO = {MQCMHO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQCMHO

```
struct tagMQCMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQCRTMH */
};
```

Deklaracja języka COBOL dla MQCMHO

```
**   MQCMHO structure
  10 MQCMHO.
```

```
**    Structure identifier
   15 MQCMHO-STRUCID        PIC X(4).
**    Structure version number
   15 MQCMHO-VERSION        PIC S9(9) BINARY.
**    Options that control the action of MQCRTMH
   15 MQCMHO-OPTIONS        PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQCMHO

```
dcl
 1 MQCMHO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQCRTMH */
```

Deklaracja High Level Assembler dla MQCMHO

```
MQCMHO                DSECT
MQCMHO_STRUCID        DS   CL4   Structure identifier
MQCMHO_VERSION        DS   F     Structure version number
MQCMHO_OPTIONS        DS   F     Options that control the action of
*                                MQCRTMH
MQCMHO_LENGTH         EQU  *-MQCMHO
MQCMHO_AREA           DS   CL(MQCMHO_LENGTH)
```

StrucId (MQCHAR4) dla MQCMHO

Jest to identyfikator struktury struktury opcji tworzenia uchwytu komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQCMHO_STRUC_ID.

Wartość musi być następująca:

MQCMHO_STRUC_ID (Identyfikator struktury MQCM)
Identyfikator struktury opcji tworzenia uchwytu komunikatu.

Dla języka programowania C zdefiniowana jest również stała **MQCMHO_STRUC_ID_ARRAY** . Ma taką samą wartość jak **MQCMHO_STRUC_ID**, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQCMHO

To pole jest zawsze polem wejściowym. Wartością początkową jest MQCMHO_VERSION_1.

Jest to numer wersji struktury; wartość musi być następująca:

MQCMHO_VERSION_1
Version-1 tworzenie struktury opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej:

BIEŻĄCA_WERSJA_MQCM
Bieżąca wersja struktury opcji tworzenia uchwytu komunikatu.

Opcje (MQLONG) dla MQCMHO

To pole jest zawsze polem wejściowym. Jego wartością początkową jest MQCMHO_DEFAULT_VALIDATION.

Można podać jedną z następujących opcji:

MQCMHO_VALIDATE,

Po wywołaniu komendy **MQSETMP** w celu ustawienia właściwości w tym uchwycie komunikatu sprawdzana jest poprawność nazwy właściwości, aby upewnić się, że:

- nie zawiera niepoprawnych znaków.

- nie zaczyna się od JMS ani od usr.JMS z wyjątkiem:
  - JMSCorrelationID
  - JMSReplyTo
  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Te nazwy są zarezerwowane dla właściwości JMS .
- nie jest jednym z następujących słów kluczowych, w dowolnej kombinacji wielkich i małych liter:
  - I
  - W zakresie
  - Escape
  - FAŁSZ
  - zawiera się w
  - JEST
  - PODOBNE
  - NIE
  - NULL
  - LUB
  - PRAWDA
- nie zaczyna się Body. lub Root. (z wyjątkiem Root.MQMD).

Jeśli właściwość jest zdefiniowana jako MQ(mq. *) i nazwa jest rozpoznawana, pola deskryptora właściwości są ustawione na poprawne wartości dla właściwości. Jeśli właściwość nie zostanie rozpoznana, w polu *Support* deskryptora właściwości zostanie ustawiona wartość **MQPD_OPTIONAL**.

MQCMHO_DEFAULT_VALIDATION,

Ta wartość określa domyślny poziom sprawdzania poprawności nazw właściwości.

Domyślny poziom sprawdzania poprawności jest równoważny poziomowi określonemu przez parametr **MQCMHO_VALIDATE**.

Jest to wartość domyślna.

MQCMHO_NO_VALIDATION,

Nie jest sprawdzana poprawność nazwy właściwości. Patrz opis **MQCMHO_VALIDATE**.

Opcja domyślna: Jeśli żadna z powyższych opcji nie jest wymagana, można użyć następującej opcji:

MQCMHO_NONE

Wszystkie opcje przyjmują wartości domyślne. Ta wartość wskazuje, że nie określono żadnych innych opcji. **MQCMHO_NONE** jest pomocna w dokumentacji programu. Opcja ta nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, nie można jej wykryć.

MQCNO-opcje połączenia

Struktura MQCNO umożliwia aplikacji określenie opcji dotyczących połączenia z menedżerem kolejek. Struktura jest parametrem wejścia/wyjścia wywołania MQCONNX.

Więcej informacji na temat używania współużytkowanych uchwytów i wywołań MQCONNX zawiera sekcja Współużytkowane (niezależne od wątku) połączenia z produktem MQCONNX.

Dostępność

Struktura MQCNO jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

Wersja

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję MQCNO, ale z wartością początkową pola *Version* ustawioną na wartość MQCNO_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *Version* wymagany numer wersji.

Zestaw znaków i kodowanie

Dane w MQCNO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako IBM MQ MQI client, struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 479. Pola w MQCNO dla MQCNO

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQCNO_STRUC_ID	'CNO¬'
Wersja (numer wersji struktury)	MQCNO_VERSION_1	1
Opcje (opcje sterujące działaniem komendy MQCONNX)	MQCNO_BRAK	0
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_2.		
ClientConnPrzesunięcie (przesunięcie struktury MQCD dla połączenia klienckiego)	Brak	0
ClientConnPtr (adres struktury MQCD dla połączenia klienta)	Brak	Pusty wskaźnik lub puste bajty
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_3.		
ConnTag (znacznik połączenia menedżera kolejek)	MQCT_NONE	Wartości null
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_4.		
SSLConfigPtr (adres struktury MQSCO dla połączenia klienta)	Brak	Pusty wskaźnik lub puste bajty
SSLConfigOffset (przesunięcie struktury MQSCO dla połączenia klienckiego)	Brak	0
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_5.		

Tabela 479. Pola w MQCNO dla MQCNO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
ConnectionId (unikalny identyfikator połączenia)	Brak	Pusty wskaźnik lub puste bajty
SecurityParmsPrzesunięcie (przesunięcie struktury MQSCO dla parametrów zabezpieczeń)	Brak	Pusty wskaźnik lub puste bajty
SecurityParmsPtr (adres struktury MQSCO dla parametrów zabezpieczeń)	Brak	Pusty wskaźnik lub puste bajty
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQCNO_VERSION_6.		
Zarezerwowane (pole zastrzeżone)	Brak	Pole zarezerwowane do dopełniania struktury do granicy 64-bitowej.
CCDTUrlLength (CCDT URL)	Brak	Długość łańcucha identyfikowanego przez *CCDTUrlPtr* lub *CCDTUrlOffset*
CCDTUrlPtr (wskaźnik URL CCDT)	Brak	Wskaźnik do łańcucha zawierającego URLidentyfikujący położenie tabeli kanałów połączenia klienckiego, która ma być używana dla połączenia.
CCDTUrlOffset (przesunięcie URL tabeli CCDT)	Brak	Przesunięcie w bajtach od łańcucha zawierającego URL określający położenie tabeli kanału połączenia klienckiego, która ma być używana dla połączenia.
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_7.		
ApplName (nazwa ustawiona przez aplikację)	Brak	Nazwa ustawiona przez aplikację na potrzeby identyfikowania połączenia z menedżerem kolejek
Reserved2 (pole zastrzeżone)	Brak	Pole zarezerwowane do dopełniania struktury do granicy 64-bitowej.
V 9.3.0 V 9.3.0 **Uwaga:** Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_8.		
V 9.3.0 BalanceParms| Przesunięcie	Brak	Przesunięcie w bajtach do struktury MQBNO

Tabela 479. Pola w MQCNO dla MQCNO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
V 9.3.0 BalanceParmsPtr	Brak	Wskaźnik do położenia struktury MQBNO

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraParametr MQCNO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCNO MyCNO = {MQCNO_DEFAULT};
```

Deklaracje językowe

Uwaga: V 9.3.0 W każdej z poniższych deklaracji dwa ostatnie wiersze (`Offset of the MQBMO structure` i `Address of the location of the MQBMO structure`) zostały dodane w IBM MQ 9.2.4 dla użytkowników CD oraz w IBM MQ 9.3.0 dla użytkowników LTS.

Deklaracja C dla MQCNO

```
typedef struct tagMQCNO MQCNO;
struct tagMQCNO {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQLONG     Options;           /* Options that control the action of
                                   MQCONNX */
  MQLONG     ClientConnOffset;  /* Offset of MQCD structure for client
                                   connection */
  MQPTR      ClientConnPtr;     /* Address of MQCD structure for client
                                   connection */
  MQBYTE128  ConnTag;           /* Queue managerconnection tag */
  PMQSCO     SSLConfigPtr;      /* Address of MQSCO structure for client
                                   connection */
  MQLONG     SSLConfigOffset;   /* Offset of MQSCO structure for client
                                   connection */
  MQBYTE24   ConnectionId;      /* Unique connection identifier */
  MQLONG     SecurityParmsOffset /* Security fields */
  PMQCSP     SecurityParmsPtr   /* Security parameters */
  MQLONG     CCDTUrlLength      /* Length of string identified by Ptr or offset */
  MQLONG     CCDTUrlOffset      /* Offset in bytes to URL of client connection channel */
  PMQURL     CCDTUrlPtr         /* Address of string containing URL */
  MQBYTE4    Reserved           /* Reserved field to pad out to 64 bit boundary */
  MQCHAR28   ApplName           /* Name set by the application to identify the connection to
                                   the queue manager */
  MQBYTE4    Reserved2          /* Reserved field to pad out to 64 bit boundary */
  MQLONG     BalanceParmsOffset /* Offset of the MQBMO structure */
  PMQBMO     BalanceParmsPtr    /* Address of the location of the MQBMO structure */
};
```

Deklaracja języka COBOL dla MQCNO

```
**   MQCNO structure
  10 MQCNO.
**    Structure identifier
   15 MQCNO-STRUCID          PIC X(4).
**    Structure version number
   15 MQCNO-VERSION          PIC S9(9) BINARY.
**    Options that control the action of MQCONNX
   15 MQCNO-OPTIONS          PIC S9(9) BINARY.
**    Offset of MQCD structure for client connection
   15 MQCNO-CLIENTCONNOFFSET PIC S9(9) BINARY.
**    Address of MQCD structure for client connection
```

```
   15 MQCNO-CLIENTCONNPTR    POINTER.
**    Queue manager connection tag
   15 MQCNO-CONNTAG          PIC X(128).
**    Address of MQSCO structure for client connection
   15 MQCNO-SSLCONFIGPTR     POINTER.
**    Offset of MQSCO structure for client connection
   15 MQCNO-SSLCONFIGOFFSET  PIC S9(9) BINARY.
**    Unique connection identifier
   15 MQCNO-CONNECTIONID     PIC X(24).
**    Offset of MQCSP structure for security parameters
   15 MQCNO-SECURITYPARMSOFFSET PIC S9(9) BINARY.
**    Address of MQCSP structure for security parameters
   15 MQCNO-SECURITYPARMSPTR POINTER.
**    Length of string identified by CCDTURLOFFSET or CCDTURLPTR
   15 MQCNO-CCDTURLLENGTH
**    Pointer to a string which contains a URL, to identify the location of the client
connection channel
   15 MQCNO-CCDTURLPTR
**    Address of string which contains a URL that identifies the location of the client
connection channel table
   15 MQCNO-CCDTURLOFFSET
**    Reserved field to pad to 64 bit boundary
   15 MQCNO-RESERVED
**    Name set by the application to identify the connection to the queue manager
   15 MQCNO-APPLNAME
**    Reserved field to pad to 64 bit boundary
   15 MQCNO-RESERVED2
**    Address of the MQBMO structure
   15 MQCNO-BALANCEPARMSOFFSET
**    Pointer to the MQBMO structure
   15 MQCNO-BALANCEPARMSPTR
```

Deklaracja języka PL/I dla MQCNO

```
dcl
 1 MQCNO based,
  3 StrucId             char(4),        /* Structure identifier */
  3 Version             fixed bin(31),  /* Structure version number */
  3 Options             fixed bin(31),  /* Options that control the action
                                           of MQCONNX */
  3 ClientConnOffset    fixed bin(31),  /* Offset of MQCD structure for
                                           client connection */
  3 ClientConnPtr       pointer,        /* Address of MQCD structure for
                                           client connection */
  3 ConnTag             char(128),      /* Queue managerconnection tag */
  3 SSLConfigPtr        pointer,        /* Address of MQSCO structure for
                                           client connection */
  3 SSLConfigOffset     fixed bin(31),  /* Offset of MQSCO structure for
                                           client connection */
  3 ConnectionId        char(24),       /* Unique connection identifier
  3 SecurityParmsOffset fixed bin(31)  /* Offset of MQCSP structure for
                                           security parameters */
  3 SecurityParmsPtr    pointer,        /* Address of MQCSP structure for
                                           security parameters */
  3 CCDTUrlLength       fixed bin(31)   /* Length of string identified by CCDTUrlPtr
                                           or CCDTUrlOffset */
  3 CCDTUrlOffset       fixed bin(31)   /* Offset in bytes to URL of client connection channel */
  3 CCDTUrlPtr          pointer         /* Pointer to string containing URL */
  3 Reserved            char(4)         /* Reserved field to pad out to 64 bit boundary */
  3 ApplName            char(28)        /* Name set by the application to identify the
connection to
                                           the queue manager */
  3 Reserved2           char(4)         /* Reserved field to pad out to 64 bit boundary */
  3 BalanceParmsOffset fixed bin(31)   /* Offset of the MQBMO structure */
  3 BalanceParmsPtr     pointer         /* Address of the MQBMO structure */
```

Deklaracja High Level Assembler dla MQCNO

```
MQCNO                    DSECT
MQCNO_STRUCID            DS   CL4    Structure identifier
MQCNO_VERSION            DS   F      Structure version number
MQCNO_OPTIONS            DS   F      Options that control the action of
*                                     MQCONNX
MQCNO_CLIENTCONNOFFSET   DS   F      Offset of MQCD structure for client
*                                     connection
MQCNO_CLIENTCONNPTR      DS   F      Address of MQCD structure for client
```

```
*                                    connection
MQCNO_CONNTAG             DS   XL128  Queue manager connection tag
MQCNO_CONNECTIONID        DS   XL24   Unique connection identifier
MQCNO_SSLCONFIGOFFSET     DS   F      Offset of MQCSP structure for security
*                                    parameters
MQCNO_SSLCONFIGPTR        DS   F      Address of MQCSP structure for security
*                                    parameters
MQCNO_LENGTH              EQU  *-MQCNO
                          ORG  MQCNO
MQCNO_AREA                DS   CL(MQCNO_LENGTH)
MQCNO_CCDTURLLENGTH       DS   F      Length of string identified by CCDTURLPTR or
*                                    CCDTURLOFFSET
MQCNO_CCDTURLOFFSET       DS   F      Offset in bytes to URL of client connection channel
MQCNO_CCDTURLPTR          DS   F      Pointer to string containing URL
RESERVED                  DS   XL4    Reserved field to pad out to 64 bit boundary
APPLNAME                  DS   CL28   Name set by the application to identify the connection to
*                                    the queue manager
RESERVED2                 DS   XL4    Reserved field to pad out to 64 bit boundary
MQCNO_BALANCEPARMSOFFSET  DS   F      Offset of the MQBMO structure
MQCNO_BALANCEPARMSPTR     DS   F      Address of the MQBMO structure
```

Deklaracja Visual Basic dla MQCNO

```
Type MQCNO
  StrucId             As String*4  'Structure identifier'
  Version             As Long      'Structure version number'
  Options             As Long      'Options that control the action of'
                                   'MQCONNX'
  ClientConnOffset    As Long      'Offset of MQCD structure for client'
                                   'connection'
  ClientConnPtr       As MQPTR     'Address of MQCD structure for client'
                                   'connection'
  ConnTag             As MQBYTE128 'Queue manager connection tag'
  SSLConfigPtr        As MQPTR     'Address of MQSCO structure for client'
                                   'connection'
  SSLConfigOffset     As Long      'Offset of MQSCO structure for client'
                                   'connection'
  ConnectionId        As MQBYTE24  'Unique connection identifier'
  SecurityParmsOffset As Long      'Offset of MQCSP structure for security'
                                   'parameters'
  SecurityParmsPtr    As MQPTR     'Address of MQCSP structure for security'
                                   'parameters'
  CCDTUrlLength       As Long      'Length of string identified by CCDTUrlPtr'
                                   'or CCDTUrlOffset'
  CCDTUrlOffset       As Long      'Offset in bytes to URL of client connection channel'
  CCDTUrlPtr          As MQPTR     'Pointer to string containing URL'
  Reserved            As MQBYTE4   'Reserved field to pad out to 64 bit boundary'
  ApplName            As String*28 'Name set by the application to identify the connection to'
                                   'the queue manager'
  Reserved2           As MQBYTE4   'Reserved field to pad out to 64 bit boundary'
  BalanceParmsOffset  As Long      'Offset in bytes to MQBNO structure'
  BalanceParmsPtr     As MQPTR     'Address of MQBNO structure'
  End Type
```

Zadania pokrewne

Korzystanie z usługi MQCONNX

StrucId (MQCHAR4) dla MQCNO

Jest to identyfikator struktury struktury opcji połączenia. Jest to zawsze pole wejściowe. Jego wartością jest MQCNO_STRUC_ID.

Wartość musi być następująca:

MQCNO_STRUC_ID
Identyfikator struktury opcji połączenia.

Dla języka programowania C zdefiniowana jest również stała MQCNO_STRUC_ID_ARRAY. Ta stała ma taką samą wartość jak MQCNO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQCNO

Wersja jest zawsze polem wejściowym. Wartością początkową jest MQCNO_VERSION_1.

Wartość musi być jedną z następujących wartości:

MQCNO_VERSION_1
Version-1 connect-struktura opcji.

MQCNO_VERSION_2
Struktura opcji połączenia Version-2 .

MQCNO_VERSION_3
Struktura opcji połączenia Version-3 .

MQCNO_VERSION_4
Struktura opcji połączenia Version-4 .

MQCNO_VERSION_5
Struktura opcji połączenia Version-5 .

MQCNO_VERSION_6
Version-6 -struktura opcji połączenia.

MQCNO_VERSION_7
Struktura opcji połączenia Version-7 .

V 9.3.0 **MQCNO_VERSION_8**
Version-8 connect-options structure.

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQCNO_CURRENT_VERSION
Bieżąca wersja struktury connect-options.

Opcje (MQLONG) dla MQCNO
Opcje sterujące działaniem komendy MQCONNX.

Opcje rozliczania

Następujące opcje sterują typem rozliczania, jeśli atrybut menedżera kolejek **AccountingConnOverride** jest ustawiony na wartość MQMON_ENABLED:

MQCNO_ACCOUNTING_MQI_ENABLED,
Jeśli gromadzenie danych monitorowania jest wyłączone w definicji menedżera kolejek przez ustawienie atrybutu **MQIAccounting** na wartość MQMON_OFF, ustawienie tej flagi powoduje włączenie gromadzenia danych rozliczania MQI.

MQCNO_ACCOUNTING_MQI_WYŁĄCZONE
Jeśli gromadzenie danych monitorowania jest wyłączone w definicji menedżera kolejek przez ustawienie atrybutu **MQIAccounting** na wartość MQMON_OFF, ustawienie tej flagi spowoduje zatrzymanie gromadzenia danych rozliczania MQI.

MQCNO_ACCOUNTING_Q_ENABLED,
Jeśli gromadzenie danych rozliczeniowych dla kolejki jest wyłączone w definicji menedżera kolejek przez ustawienie atrybutu **MQIAccounting** na wartość MQMON_OFF, ustawienie tej flagi umożliwia gromadzenie danych rozliczeniowych dla kolejek, które określają menedżer kolejek w polu *MQIAccounting* swojej definicji kolejki.

MQCNO_ACCOUNTING_Q_DISABLED
Jeśli gromadzenie danych rozliczeniowych dla kolejki jest wyłączone w definicji menedżera kolejek przez ustawienie atrybutu **MQIAccounting** na wartość MQMON_OFF, ustawienie tej flagi powoduje wyłączenie gromadzenia danych rozliczeniowych dla tych kolejek, które określają menedżer kolejek w polu *MQIAccounting* swojej definicji kolejki.

Jeśli żadna z tych opcji nie jest zdefiniowana, rozliczanie połączenia odbywa się zgodnie z definicją w atrybutach menedżera kolejek.

Opcje powiązania

Następujące opcje sterują typem powiązania IBM MQ , które ma być używane. Należy podać tylko jedną z następujących opcji:

MQCNO_STANDARD_BINDING
Aplikacja i agent lokalnego menedżera kolejek (komponent zarządzający operacjami kolejkowania) działają w oddzielnych jednostkach wykonywania (zwykle w oddzielnych procesach). Taki układ zapewnia integralność menedżera kolejek, czyli chroni menedżera kolejek przed błędnym programem.

Jeśli menedżer kolejek obsługuje wiele typów powiązań, a użytkownik ustawi opcję MQCNO_STANDARD_BINDING, menedżer kolejek używa atrybutu **DefaultBindType** w sekcji `Connection` pliku `qm.ini` , aby wybrać rzeczywisty typ powiązania. Jeśli ta sekcja nie jest zdefiniowana lub wartość nie może być użyta albo nie jest odpowiednia dla aplikacji, menedżer kolejek wybiera odpowiedni typ powiązania. Menedżer kolejek ustawia rzeczywisty typ powiązania używany w opcjach połączenia.

Użyj opcji MQCNO_STANDARD_BINDING w sytuacjach, w których aplikacja mogła nie zostać w pełni przetestowana lub może być niepewna lub niewiarygodna. Wartością domyślną jest MQCNO_STANDARD_BINDING.

Ta opcja jest obsługiwana we wszystkich środowiskach.

Jeśli tworzone jest dowiązanie do biblioteki `mqm` , najpierw podejmowana jest próba nawiązania standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Jeśli załadowanie bazowej biblioteki serwera nie powiodło się, zamiast tego podejmowana jest próba nawiązania połączenia z klientem.

- Aby zmienić zachowanie MQCONN (lub MQCONNX, jeśli określono opcję MQCNO_STANDARD_BINDING), należy ustawić zmienną środowiskową MQ_CONNECT_TYPE na jedną z następujących opcji. Należy zauważyć, że jest to wyjątek: jeśli parametr MQCNO_FASTPATH_BINDING zostanie określony z parametrem MQ_CONNECT_TYPE ustawionym na wartość LOCAL lub STANDARD, połączenia krótkiej ścieżki mogą zostać zdegradowane przez administratora bez zmiany pokrewnej w aplikacji.

Tabela 480. Wartości dla MQ_CONNECT_TYPE, które zmieniają zachowanie MQCONN lub MQCONNX

Wartość	**Znaczenie**
KLIENT	Podejmowana jest tylko próba nawiązania połączenia z klientem.
Krótka ścieżka	Ta wartość była obsługiwana w poprzednich wersjach, ale teraz jest ignorowana, jeśli zostanie podana.
LOKALNA	Podejmowana jest tylko próba nawiązania połączenia z serwerem. Połączenia krótkiej ścieżki są obniżane do standardowego połączenia z serwerem.
STANDARDOWA	Obsługiwane ze względu na kompatybilność z poprzednimi wersjami. Ta wartość jest teraz traktowana jako LOCAL.

- Jeśli zmienna środowiskowa MQ_CONNECT_TYPE nie jest ustawiona podczas wywołania MQCONNX, podejmowana jest próba nawiązania standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. Jeśli załadowanie biblioteki serwera nie powiedzie się, podejmowana jest próba nawiązania połączenia z klientem.

MQCNO_FASTPATH_BINDING,
Aplikacja i agent lokalnego menedżera kolejek są częścią tej samej jednostki wykonywania. Jest to przeciwieństwo typowej metody wiązania, w której aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania.

Opcja MQCNO_FASTPATH_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

Zaletą opcji MQCNO_FASTPATH_BINDING może być sytuacja, w której wiele procesów zużywa więcej zasobów niż cały zasób używany przez aplikację.Aplikacja używająca powiązania krótkiej ścieżki jest nazywana *aplikacją zaufaną*.

Przy podejmowaniu decyzji, czy użyć powiązania krótkiej ścieżki, należy wziąć pod uwagę następujące ważne kwestie:

- Użycie opcji MQCNO_FASTPATH_BINDING nie zapobiega zmianie lub uszkodzeniu komunikatów aplikacji i innych obszarów danych należących do menedżera kolejek. Tej opcji należy używać tylko w sytuacjach, w których te problemy zostały w pełni przeanalizowane.
- Aplikacja nie może używać asynchronicznych sygnałów ani przerwań zegara (takich jak `sigkill`) z powiązaniem MQCNO_FASTPATH_BINDING. Istnieją również ograniczenia dotyczące używania segmentów pamięci współużytkowanej.
- Aplikacja musi użyć wywołania MQDISC, aby rozłączyć się z menedżerem kolejek.
- Aplikacja musi zostać zakończona przed zakończeniem działania menedżera kolejek za pomocą komendy `endmqm` .
- IBM i W systemie IBM izadanie musi być uruchomione w profilu użytkownika należącego do grupy `QMQMADM` . Ponadto program nie może zostać zatrzymany nieprawidłowo, w przeciwnym razie mogą wystąpić nieprzewidywalne rezultaty.
- Linux AIX W systemie AIX and Linuxidentyfikator użytkownika `mqm` musi być efektywnym identyfikatorem użytkownika, a identyfikator grupy `mqm` musi być efektywnym identyfikatorem grupy. Aby aplikacja była uruchamiana w ten sposób, należy skonfigurować program w taki sposób, aby jego właścicielem był identyfikator użytkownika `mqm` i identyfikator grupy `mqm` , a następnie ustawić w programie bity uprawnień `setuid` i `setgid` .

  IBM MQ Object Authority Manager (OAM) nadal używa rzeczywistego identyfikatora użytkownika do sprawdzania uprawnień.
- Windows W systemie Windowsprogram musi należeć do grupy `mqm` . Powiązanie krótkiej ścieżki nie jest obsługiwane w przypadku aplikacji 64-bitowych.

Opcja MQCNO_FASTPATH_BINDING jest obsługiwana w następujących środowiskach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

z/OS W systemie z/OSopcja jest akceptowana, ale ignorowana.

Więcej informacji na temat wpływu używania zaufanych aplikacji zawiera sekcja Ograniczenia dotyczące zaufanych aplikacji.

MQCNO_SHARED_BINDING

Z opcją MQCNO_SHARED_BINDING aplikacja i agent lokalnego menedżera kolejek współużytkują pewne zasoby. Opcja MQCNO_SHARED_BINDING jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

MQCNO_IZOLOWANE_POWIĄZANIE

W tym przypadku proces aplikacji i agent lokalnego menedżera kolejek są izolowane od siebie, ponieważ nie współużytkują zasobów. Opcja MQCNO_ISOLATED_BINDING jest ignorowana, jeśli

menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

MQCNO_CLIENT_BINDING

Podaj tę opcję, aby aplikacja próbowała nawiązać tylko połączenie z klientem. Ta opcja ma następujące ograniczenia:

- z/OS Opcja MQCNO_CLIENT_BINDING jest ignorowana w systemie z/OS.
- Opcja MQCNO_CLIENT_BINDING jest odrzucana z opcją MQRC_OPTIONS_ERROR, jeśli została określona z dowolną opcją powiązania MQCNO inną niż MQCNO_STANDARD_BINDING.
- Opcja MQCNO_CLIENT_BINDING nie jest dostępna dla produktu Java lub .NET , ponieważ mają one własne mechanizmy wyboru typu powiązania.

POWIĄZANIE_LOKALNE_MQCNO_BINDING

Podaj tę opcję, aby aplikacja próbowała nawiązać połączenie z serwerem. Jeśli określono również opcję MQCNO_FASTPATH_BINDING, MQCNO_ISOLATED_BINDING lub MQCNO_SHARED_BINDING, to połączenie jest tego typu i zostało opisane w tej sekcji. W przeciwnym razie zostanie podjęta próba nawiązania standardowego połączenia z serwerem przy użyciu domyślnego typu powiązania. MQCNO_LOCAL_BINDING ma następujące ograniczenia:

- z/OS Opcja MQCNO_LOCAL_BINDING jest ignorowana w systemie z/OS.
- Opcja MQCNO_LOCAL_BINDING została odrzucona z opcją MQRC_OPTIONS_ERROR, jeśli została określona z dowolną opcją ponownego połączenia MQCNO inną niż MQCNO_RECONNECT_AS_DEF.
- Opcja MQCNO_LOCAL_BINDING nie jest dostępna dla produktu Java lub .NET , ponieważ mają one własne mechanizmy wyboru typu powiązania.

Na następujących platformach do sterowania typem używanego powiązania można użyć zmiennej środowiskowej `MQ_CONNECT_TYPE` z typem powiązania określonym w polu `Options` .

- AIX AIX
- Linux Linux
- Windows Windows

W przypadku określenia tej zmiennej środowiskowej musi ona mieć wartość FASTPATH lub STANDARD ; jeśli ma inną wartość, jest ignorowana. W wartości zmiennej środowiskowej rozróżniana jest wielkość liter. Więcej informacji na ten temat zawiera sekcja Zmienna środowiskowa MQCONNX .

Zmienna środowiskowa i pole *`Options`* współdziałają ze sobą w następujący sposób:

- Jeśli zmienna środowiskowa zostanie pominięta lub zostanie jej nadana wartość, która nie jest obsługiwana, użycie powiązania krótkiej ścieżki jest określane wyłącznie przez pole `Options` .
- Jeśli zmiennej środowiskowej zostanie nadana obsługiwana wartość, powiązanie krótkiej ścieżki będzie używane tylko wtedy, gdy zarówno zmienna środowiskowa, jak i pole `Options` określają powiązanie krótkiej ścieżki.

Opcje znacznika połączenia

Te opcje są obsługiwane tylko podczas nawiązywania połączenia z menedżerem kolejek produktu z/OS i sterują użyciem znacznika połączenia `ConnTag`. Można określić tylko jedną z tych opcji.

Dokładna implementacja znaczników połączenia różni się w systemach IBM MQ for z/OS i IBM MQ for Multiplatforms:

- z/OS Poniższe opcje, oprócz opcji *MQCNO_GENERATE_CONN_TAG*, są obsługiwane tylko podczas nawiązywania połączenia z menedżerem kolejek produktu z/OS i sterują użyciem znacznika połączenia. Można podać tylko jedną z obsługiwanych opcji.

- ALW Opcja *MQCNO_GENERATE_CONN_TAG* jest obsługiwana tylko na platformach innych niż z/OS.

ALW **MQCNO_GENERATE_CONN_TAG**

Zwraca znacznik połączenia powiązany z tym połączeniem przez menedżer kolejek w wyjściowej strukturze MQCNO.

Zwrócony znacznik połączenia będzie identyczny dla wszystkich połączeń, które menedżer kolejek traktuje jako pojedynczą instancję aplikacji.

z/OS **MQCNO_SERIALIZE_CONN_TAG_Q_MGR**

Ta opcja żąda wyłącznego użycia znacznika połączenia w menedżerze kolejek lokalnych. Jeśli znacznik połączenia jest już używany w lokalnym menedżerze kolejek, wywołanie MQCONNX kończy się niepowodzeniem z kodem przyczyny MQRC_CONN_TAG_IN_USE. Na wynik wywołania nie ma wpływu użycie znacznika połączenia w innym miejscu w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych.

z/OS **MQCNO_SERIALIZE_CONN_TAG_QSG**

Ta opcja żąda wyłącznego użycia znacznika połączenia w grupie współużytkowania kolejek, do której należy lokalny menedżer kolejek. Jeśli znacznik połączenia jest już używany w grupie współużytkowania kolejek, wywołanie MQCONNX kończy się niepowodzeniem z kodem przyczyny MQRC_CONN_TAG_IN_USE.

z/OS **MQCNO_RESTRICT_CONN_TAG_Q_MGR (mqcno_restrict_conn_q_mgr)**

Ta opcja żąda współużytkowanego użycia znacznika połączenia w menedżerze kolejek lokalnych. Jeśli znacznik połączenia jest już używany w lokalnym menedżerze kolejek, wywołanie MQCONNX może zakończyć się powodzeniem, jeśli aplikacja żądająca jest uruchomiona w tym samym zasięgu przetwarzania, co istniejący użytkownik znacznika. Jeśli ten warunek nie jest spełniony, wywołanie MQCONNX kończy się niepowodzeniem z kodem przyczyny MQRC_CONN_TAG_IN_USE. Na wynik wywołania nie ma wpływu użycie znacznika połączenia w innym miejscu w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych.

- Aplikacje muszą działać w tej samej przestrzeni adresowej MVS, aby współużytkować znacznik połączenia. Jeśli aplikacja używająca znacznika połączenia jest aplikacją kliencką, parametr MQCNO_RESTRICT_CONN_TAG_Q_MGR nie jest dozwolony.

z/OS **MQCNO_RESTRICT_CONN_TAG_QSG**

Ta opcja żąda współużytkowanego użycia znacznika połączenia w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych. Jeśli znacznik połączenia jest już używany w grupie współużytkowania kolejek, wywołanie MQCONNX może zakończyć się pomyślnie, pod warunkiem, że aplikacja żądająca działa w tym samym zasięgu przetwarzania i jest połączona z tym samym menedżerem kolejek, co istniejący użytkownik znacznika.

Jeśli te warunki nie są spełnione, wywołanie MQCONNX kończy się niepowodzeniem z kodem przyczyny MQRC_CONN_TAG_IN_USE.

- Aplikacje muszą działać w tej samej przestrzeni adresowej MVS, aby współużytkować znacznik połączenia. Jeśli aplikacja używająca znacznika połączenia jest aplikacją kliencką, opcja MQCNO_RESTRICT_CONN_TAG_QSG nie jest dozwolona.

Jeśli żadna z tych opcji nie zostanie podana, nie będzie używana opcja `ConnTag` . Te opcje nie są poprawne, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_3.

Opcje współużytkowania uchwytu

Multi

Te opcje są obsługiwane w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

Sterują one współużytkowaniem uchwytów między różnymi wątkami (jednostkami przetwarzania równoległego) w ramach tego samego procesu. Można określić tylko jedną z następujących opcji:

MQCNO_HANDLE_SHARE_NONE

Ta opcja wskazuje, że uchwyty połączenia i obiektu mogą być używane tylko przez wątek, który spowodował przydzielenie uchwytu (czyli wątek, który wywołał wywołanie MQCONN, MQCONNX lub MQOPEN). Uchwyty nie mogą być używane przez inne wątki należące do tego samego procesu.

MQCNO_HANDLE_SHARE_BLOCK

Ta opcja wskazuje, że połączenia i uchwyty obiektów przydzielone przez jeden wątek procesu mogą być używane przez inne wątki należące do tego samego procesu. Jednak tylko jeden wątek na raz może używać konkretnego uchwytu, co oznacza, że dozwolone jest tylko użycie uchwytu szeregowego. Jeśli wątek próbuje użyć uchwytu, który jest już używany przez inny wątek, wywołanie blokuje (oczekuje) aż uchwyt stanie się dostępny.

MQCNO_HANDLE_SHARE_NO_BLOCK

Jest to taka sama sytuacja, jak w przypadku tabeli MQCNO_HANDLE_SHARE_BLOCK, z tą różnicą, że jeśli uchwyt jest używany przez inny wątek, wywołanie jest natychmiast kończone z MQCC_FAILED i MQRC_CALL_IN_PROGRESS zamiast blokowania do momentu, gdy uchwyt stanie się dostępny.

Wątek może mieć zero lub jeden niewspółużytkowany uchwyt:

- Każde wywołanie MQCONN lub MQCONNX, które określa parametr MQCNO_HANDLE_SHARE_NONE, zwraca nowy niewspółużytkowany uchwyt w pierwszym wywołaniu i ten sam niewspółużytkowany uchwyt w drugim i późniejszym wywołaniu (przy założeniu, że nie ma wywołania MQDISC). Kod przyczyny to MQRC_ALREADY_CONNECTED dla drugiego i późniejszych wywołań.
- Każde wywołanie MQCONNX, które określa MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, zwraca nowy współużytkowany uchwyt dla każdego wywołania.

Uchwyty obiektów dziedziczą te same właściwości współużytkowania, co uchwyt połączenia określony w wywołaniu MQOPEN, które utworzyło uchwyt obiektu. Ponadto jednostki pracy dziedziczą te same właściwości współużytkowania, co uchwyt połączenia używany do uruchamiania jednostki pracy. Jeśli jednostka pracy jest uruchamiana w jednym wątku przy użyciu uchwytu współużytkowanego, jednostka pracy może być aktualizowana w innym wątku przy użyciu tego samego uchwytu.

Jeśli opcja współużytkowania uchwytu nie zostanie określona, środowisko będzie określać wartość domyślną:

- Windows W środowisku Microsoft Transaction Server (MTS) wartością domyślną jest MQCNO_HANDLE_SHARE_BLOCK.
- W innych środowiskach wartość domyślna jest taka sama jak wartość parametru MQCNO_HANDLE_SHARE_NONE.

Opcje ponownego połączenia

Opcje ponownego połączenia określają, czy połączenie można ponownie nawiązać. Można ponownie nawiązywać połączenia tylko z klientami.

MQCNO_RECONNECT_AS_DEF

Opcja ponownego połączenia jest tłumaczona na wartość domyślną. Jeśli nie ustawiono wartości domyślnej, wartość tej opcji jest ustawiana na DISABLED. Wartość opcji jest przekazywana do serwera i może być odpytywana przez PCF i MQSC.

MQCNO_RECONNECT

Aplikacja może zostać ponownie połączona z dowolnym menedżerem kolejek zgodnie z wartością parametru **QmgrName** parametru MQCONNX. Opcji MQCNO_RECONNECT należy używać tylko wtedy, gdy nie ma powinowactwa między aplikacją kliencką a menedżerem kolejek, z którym początkowo nawiązywane jest połączenie. Wartość opcji jest przekazywana do serwera i może być odpytywana przez PCF i MQSC.

MQCNO_RECONNECT_DISABLED

Nie można ponownie połączyć aplikacji. Wartość opcji nie jest przekazywana do serwera.

MQCNO_RECONNECT_Q_MGR

Aplikację można ponownie połączyć tylko z menedżerem kolejek, z którym pierwotnie była połączona. Tej wartości należy użyć, jeśli można ponownie nawiązać połączenie z klientem, ale istnieje powinowactwo między aplikacją kliencką a menedżerem kolejek, z którym pierwotnie nawiązano połączenie. Wartość tę należy wybrać, jeśli klient ma automatycznie nawiązywać ponowne połączenie z instancją rezerwową menedżera kolejek o wysokiej dostępności. Wartość opcji jest przekazywana do serwera i może być odpytywana przez PCF i MQSC.

Opcji MQCNO_RECONNECT, MQCNO_RECONNECT_DISABLED i MQCNO_RECONNECT_Q_MGR należy używać tylko dla połączeń klienckich. Jeśli opcje są używane dla połączenia powiązania, operacja MQCONNX kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_OPTIONS_ERROR. Automatyczne ponowne łączenie klienta nie jest obsługiwane przez IBM MQ classes for Java

Opcje współużytkowania konwersacji

Poniższe opcje mają zastosowanie tylko do połączeń klienckich TCP/IP. W przypadku kanałów SNA, SPX i NetBios wartości te są ignorowane, a kanał działa tak, jak w poprzednich wersjach produktu.

MQCNO_NO_CONV_SHARING,

Ta opcja nie zezwala na współużytkowanie konwersacji.

Funkcji MQCNO_NO_CONV_SHARING można użyć w sytuacjach, w których konwersacje są mocno obciążone, a więc w sytuacji, gdy rywalizacja jest możliwa na końcu połączenia z serwerem instancji kanału, w której istnieją konwersacje współużytkowania. MQCNO_NO_CNV_SHARING zachowuje się jak SHARECNV (1), gdy jest połączony z kanałem, który obsługuje współużytkowanie konwersacji, i SHARECNV (0), gdy jest połączony z kanałem, który nie obsługuje współużytkowania konwersacji.

MQCNO_ALL_CONVS_SHARE

Ta opcja umożliwia współużytkowanie konwersacji; aplikacja nie ogranicza liczby połączeń w instancji kanału. Jest to wartość domyślna.

Jeśli aplikacja wskazuje, że instancja kanału może współużytkować, ale definicja *SharingConversations* (SHARECNV) na końcu połączenia z serwerem kanału jest ustawiona na jeden, współużytkowanie nie występuje i nie jest wyświetlane ostrzeżenie dla aplikacji.

Podobnie, jeśli aplikacja wskazuje, że współużytkowanie jest dozwolone, ale definicja połączenia z serwerem *SharingConversations* jest ustawiona na zero, nie jest wyświetlane żadne ostrzeżenie, a aplikacja zachowuje się tak samo jak klient w wersjach wcześniejszych niż IBM WebSphere MQ 7.0; ustawienie aplikacji dotyczące współużytkowania konwersacji jest ignorowane.

Opcje MQCNO_NO_CONV_SHARING i MQCNO_ALL_CONVS_SHARE wykluczają się wzajemnie. Jeśli obie opcje są określone dla konkretnego połączenia, połączenie jest odrzucane z kodem przyczyny MQRC_OPTIONS_ERROR.

Opcje definicji kanału

Następujące opcje sterują użyciem struktury definicji kanału przekazanej w MQCNO:

MQCNO_CD_FOR_OUTPUT_ONLY,

Ta opcja zezwala, aby struktura definicji kanału w MQCNO była używana tylko do zwracania nazwy kanału użytej w pomyślnym wywołaniu MQCONNX.

Jeśli nie podano poprawnej struktury definicji kanału, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CD_ERROR.

Jeśli aplikacja nie działa jako klient, opcja jest ignorowana.

Zwrócona nazwa kanału może zostać użyta w kolejnym wywołaniu MQCONNX za pomocą opcji MQCNO_USE_CD_SELECTION w celu ponownego nawiązania połączenia przy użyciu tej samej definicji kanału. Może to być przydatne, gdy w tabeli kanału klienta istnieje wiele odpowiednich definicji kanałów.

MQCNO_USE_CD_SELECTION

Ta opcja umożliwia wywołanie MQCONNX w celu nawiązania połączenia przy użyciu nazwy kanału zawartej w strukturze definicji kanału przekazanej w MQCNO.

Jeśli zmienna środowiskowa MQSERVER jest ustawiona, używana jest definicja kanału zdefiniowana przez tę zmienną. Jeśli opcja MQSERVER nie jest ustawiona, używana jest tabela kanałów klienta.

Jeśli definicja kanału ze zgodną nazwą kanału i nazwą menedżera kolejek nie zostanie znaleziona, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_Q_MGR_NAME_ERROR.

Jeśli nie podano poprawnej struktury definicji kanału, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CD_ERROR.

Jeśli aplikacja nie działa jako klient, opcja jest ignorowana.

Opcja domyślna

Jeśli nie jest wymagana żadna z powyższych opcji, można użyć następującej opcji:

MQCNO_BRAK

Nie określono żadnych opcji.

Użyj opcji MQCNO_NONE, aby wspomóc dokumentację programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją MQCNO_ *, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

ClientConn-przesunięcie (MQLONG) dla MQCNO

ClientConnPrzesunięcie jest przesunięciem w bajtach struktury definicji kanału MQCD od początku struktury MQCNO. Przesunięcie może być dodatnie lub ujemne. To pole jest polem wejściowym o wartości początkowej 0.

Parametru *ClientConnOffset* należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client. Informacje na temat sposobu użycia tego pola zawiera opis pola *ClientConnPtr* .

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_2.

ClientConnPtr (MQPTR) dla MQCNO

ClientConnPtr jest polem wejściowym. Jego wartością początkową jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Opcji *ClientConnOffset* i *ClientConnPtr* należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX działa jako IBM MQ MQI client. Określając jedno lub inne z tych pól, aplikacja może sterować definicją kanału połączenia klienckiego, udostępniając strukturę definicji kanału MQCD zawierającą wymagane wartości.

Jeśli aplikacja działa jako aplikacja IBM MQ MQI client, ale nie udostępnia struktury MQCD, do wybrania definicji kanału jest używana zmienna środowiskowa MQSERVER . Jeśli parametr MQSERVER nie jest ustawiony, używana jest tabela kanałów klienta.

Jeśli aplikacja nie jest uruchomiona jako IBM MQ MQI client, wartości *ClientConnOffset* i *ClientConnPtr* są ignorowane.

Jeśli aplikacja udostępnia strukturę MQCD, należy ustawić wymienione pola na wymagane wartości. Inne pola w strukturze MQCD są ignorowane. Łańcuchy znaków można dopełnić spacjami do długości pola lub zakończyć je znakiem o kodzie zero. Więcej informacji na temat pól w strukturze MQCD zawiera sekcja "Pola" na stronie 1527 .

Tabela 481. Pola w MQCD

Pole w MQCD	Wartość
ChannelName	Nazwa kanału.
Version	Numer wersji struktury. Wartość nie może być mniejsza niż MQCD_VERSION_7.
TransportType	Dowolny obsługiwany typ transportu.
ModeName	Nazwa trybu jednostki logicznej 6.2 .
TpName	Nazwa programu transakcyjnego jednostki logicznej 6.2 .
SecurityExit	Nazwa wyjścia zabezpieczeń kanału.
SendExit	Nazwa wyjścia wysyłania kanału.
ReceiveExit	Nazwa wyjścia odbierania kanału.
MaxMsgLength	Maksymalna długość (w bajtach) komunikatów, które mogą być wysyłane przez kanał połączenia klienta.
SecurityUserData	Dane użytkownika dla wyjścia zabezpieczeń.
SendUserData	Dane użytkownika dla wyjścia wysyłania.
ReceiveUserData	Dane użytkownika dla wyjścia odbierania.
UserIdentifier	Identyfikator użytkownika, który ma być używany do ustanawiania sesji LU 6.2 .
Password	Hasło, które ma być używane do nawiązywania sesji LU 6.2 .
ConnectionName	Nazwa połączenia.
HeartbeatInterval	Czas w sekundach między przepływami pulsu.
StrucLength	Długość struktury MQCD.
ExitNameLength	Długość nazw wyjść adresowanych przez *SendExitPtr* i *ReceiveExitPtr*. Wartość musi być większa od zera, jeśli parametr *SendExitPtr* lub *ReceiveExitPtr* jest ustawiony na wartość, która nie jest wskaźnikiem pustym.

Tabela 481. Pola w MQCD (kontynuacja)

Pole w MQCD	Wartość
ExitDataLength	Długość danych wyjścia adresowana przez *SendUserDataPtr* i *ReceiveUserDataPtr*. Wartość musi być większa od zera, jeśli parametr *SendUserDataPtr* lub *ReceiveUserDataPtr* jest ustawiony na wartość, która nie jest wskaźnikiem pustym.
SendExitsDefined	Liczba wyjść wysyłania zaadresowanych przez *SendExitPtr*. Jeśli ma wartość zero, *SendExit* i *SendUserData* podają nazwę wyjścia i dane. Jeśli wartość jest większa od zera, *SendExitPtr* i *SendUserDataPtr* podaj nazwy i dane wyjścia, a pola *SendExit* i *SendUserData* muszą być puste.
ReceiveExitsDefined	Liczba wyjść odbierania zaadresowanych przez *ReceiveExitPtr*. Jeśli ma wartość zero, *ReceiveExit* i *ReceiveUserData* podają nazwę wyjścia i dane. Jeśli wartość jest większa od zera, *ReceiveExitPtr* i *ReceiveUserDataPtr* podaj nazwy i dane wyjścia, a pola *ReceiveExit* i *ReceiveUserData* muszą być puste.
SendExitPtr	Adres pierwszego wyjścia wysyłania.
SendUserDataPtr	Adres danych dla pierwszego wyjścia wysyłania.
ReceiveExitPtr	Adres pierwszego wyjścia odbierania.
ReceiveUserDataPtr	Adres danych dla pierwszego wyjścia odbierania.
LongRemoteUserIdLength	Długość długiego identyfikatora użytkownika zdalnego.
LongRemoteUserIdPtr	Adres długiego identyfikatora użytkownika zdalnego.
RemoteSecurityId	Identyfikator ochrony zdalnej.
SSLCipherSpec	TLS CipherSpec.
SSLPeerNamePtr	Adres nazwy węzła TLS.
SSLPeerNameLength	Długość nazwy węzła TLS.
KeepAliveInterval	Wartość przekazana do stosu komunikacji dla czasu podtrzymywania połączenia dla kanału
LocalAddress	Lokalny adres komunikacji, w tym adres IP adaptera sieci lokalnej, który ma być używany, oraz zakres portów, które mają być używane dla połączeń wychodzących.

Udostępnij strukturę definicji kanału na jeden z dwóch sposobów:

- Używając pola przesunięcia *ClientConnOffset*

  W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą strukturę MQCNO, po której następuje struktura definicji kanału MQCD, i ustawić parametr *ClientConnOffset* na przesunięcie struktury definicji kanału od początku struktury MQCNO. Upewnij się, że to przesunięcie jest poprawne. Parametr *ClientConnPtr* musi być ustawiony na wskaźnik pusty lub bajty puste.

  Języka *ClientConnOffset* należy używać w przypadku języków programowania, które nie obsługują typu danych wskaźnika lub implementują typ danych wskaźnika w sposób, który nie jest przenośny w różnych środowiskach (na przykład w języku programowania COBOL).

  W języku programowania Visual Basic: struktura złożona o nazwieKomenda MQCNOCD jest udostępniana w pliku nagłówkowym CMQXB.BAS-ta struktura zawiera strukturę MQCNO, po której następuje struktura MQCD. Zainicjuj MQCNOCD, wywołując podprocedurę MQCNOCD_DEFAULTS. Komenda MQCNOCD jest używana zMQCONNXDowolny wariant wywołania MQCONNX. Więcej szczegółów zawiera opis wywołania MQCONNX.
- Za pomocą pola wskaźnika *ClientConnPtr*

W takim przypadku aplikacja może zadeklarować strukturę definicji kanału niezależnie od struktury MQCNO i ustawić parametr *ClientConnPtr* na adres struktury definicji kanału. Ustaw *ClientConnOffset* na zero.

ClientConnPtr służy do obsługi języków programowania obsługujących typ danych wskaźnika w sposób przenośny w różnych środowiskach (na przykład w języku programowania C).

W języku programowania C można użyć zmiennej makra MQCD_CLIENT_CONN_DEFAULT, aby udostępnić wartości początkowe dla struktury, które są bardziej odpowiednie do użycia w wywołaniu MQCONNX niż wartości początkowe udostępniane przez MQCD_DEFAULT.

Niezależnie od wybranej techniki można użyć tylko jednej z następujących metod: *ClientConnOffset* i *ClientConnPtr* ; Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CLIENT_CONN_ERROR, jeśli oba są niezerowe.

Po zakończeniu wywołania MQCONNX struktura MQCD nie jest ponownie przywoływana.

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_2.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, to pole jest deklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null.

ConnTag (MQBYTE128) dla MQCNO na wielu platformach

Znacznik połączenia jest koncepcyjnie podobny do identyfikatora połączenia, ale może obejmować wiele pokrewnych połączeń, identyfikując je jako pojedynczą instancję aplikacji. W przypadku wielu platform znacznik połączenia jest generowany przez menedżer kolejek w czasie połączenia.

Więcej informacji na ten temat zawiera sekcja identyfikator połączenia i instancja aplikacji.

Wygenerowane znaczniki połączenia są czytelne dla człowieka (`semi`). Oznacza to, że można je wyświetlać i filtrować za pomocą komend MQSC, tak jak w przypadku łańcuchów w lokalnym zestawie znaków. Połączeniom, które są znane programowi IBM MQ jako pokrewne, automatycznie przypisywany jest ten sam znacznik połączenia. To przypisanie jest szczególnie ważne dla równoważenia aplikacji.

Wygenerowany znacznik połączenia jest widoczny na trzy sposoby:

- W wyjściowej strukturze MQCNO w wywołaniu MQCONNX, jeśli określono wartość MQCNO_GENERATE_CONN_TAG .
- W danych wyjściowych komendy DISPLAY CONN (lub odpowiedników programowych).
- W danych wyjściowych komendy DISPLAY APSTATUS (lub jej odpowiedników).

Znacznik przestaje być poprawny, gdy aplikacja kończy działanie lub wysyła wywołanie MQDISC.

Odsyłacze pokrewne

“ConnTag (MQBYTE128) dla MQCNO w systemie IBM MQ for z/OS” na stronie 337
Znacznik połączenia jest koncepcyjnie podobny do identyfikatora połączenia, ale może obejmować wiele pokrewnych połączeń, identyfikując je jako pojedynczą instancję aplikacji. W systemie IBM MQ for z/OSznacznik połączenia jest polem wejściowym udostępnianym przez aplikację i używanym w połączeniu z opcjami MQCNO_ * _CONN_TAG w celu przekształcenia do postaci szeregowej połączeń z tej instancji aplikacji.

z/OS *ConnTag (MQBYTE128) dla MQCNO w systemie IBM MQ for z/OS*

Znacznik połączenia jest koncepcyjnie podobny do identyfikatora połączenia, ale może obejmować wiele pokrewnych połączeń, identyfikując je jako pojedynczą instancję aplikacji. W systemie IBM MQ for z/OSznacznik połączenia jest polem wejściowym udostępnianym przez aplikację i używanym w połączeniu z opcjami MQCNO_ * _CONN_TAG w celu przekształcenia do postaci szeregowej połączeń z tej instancji aplikacji.

Jeśli istnieje wiele instancji aplikacji, które mają być jednocześnie połączone, każda z nich musi dostarczyć unikalną wartość dla tego pola. Więcej szczegółów zawierają opisy tych opcji znaczników połączenia .

Uwagi:

- W systemie IBM MQ for z/OSnie ma możliwości administracyjnego określenia znacznika połączenia powiązanego z aplikacją w czasie wykonywania.
- Wartości znaczników połączenia rozpoczynające się od MQ wielkimi, małymi lub mieszanymi literami w kodzie ASCII lub EBCDIC są zastrzeżone dla produktów IBM . Nie należy używać wartości znaczników połączenia rozpoczynających się od tych liter.

Jeśli znacznik nie jest wymagany, należy użyć następującej wartości specjalnej:

MQCT_NONE
Wartością długości pola jest zero binarne.

W języku programowania C zdefiniowana jest również stała MQCT_NONE_ARRAY. Ta stała ma taką samą wartość jak MQCT_NONE, ale jest tablicą znaków zamiast łańcucha.

Pole ConnTag jest używane podczas nawiązywania połączenia z menedżerem kolejek produktu z/OS .

Długość tego pola jest określona przez wartość MQ_CONN_TAG_LENGTH. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_3.

Multi Więcej informacji na temat używania znacznika połączenia w systemie IBM MQ for Multiplatformszawiera sekcja "ConnTag (MQBYTE128) dla MQCNO na wielu platformach" na stronie 337 .

SSLConfigPtr (PMQSCO) dla MQCNO

SSLConfigPtr jest polem wejściowym. Jego wartością początkową jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Parametrów *SSLConfigPtr* i *SSLConfigOffset* należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client , a protokołem kanału jest TCP/IP. Jeśli aplikacja nie jest uruchomiona jako klient IBM MQ lub jeśli protokół kanału nie jest protokołem TCP/IP, wartości *SSLConfigPtr* i *SSLConfigOffset* są ignorowane.

Określenie wartości *SSLConfigPtr* lub *SSLConfigOffset*oraz wartości *ClientConnPtr* lub *ClientConnOffset*powoduje, że aplikacja może sterować użyciem protokołu TLS dla połączenia klienckiego. Jeśli informacje o protokole TLS są określane w ten sposób, zmienne środowiskowe MQSSLKEYR i MQSSLCRYP są ignorowane. Informacje dotyczące protokołu TLS w tabeli definicji kanału klienta (CCDT) są również ignorowane.

Informacje o protokole TLS można podać tylko w następujących sytuacjach:

- Pierwsze wywołanie MQCONNX procesu klienta lub
- Kolejne wywołanie MQCONNX, gdy wszystkie poprzednie połączenia TLS z menedżerem kolejek zostały zakończone przy użyciu MQDISC.

Są to jedyne stany, w których można zainicjować środowisko TLS dla całego procesu. Jeśli wywołanie MQCONNX zostało wydane z podaniem informacji TLS, gdy środowisko TLS już istnieje, informacje TLS w wywołaniu są ignorowane, a połączenie jest nawiązywane przy użyciu istniejącego środowiska TLS. W tym przypadku wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SSL_ALREADY_INITIALIZED.

Strukturę MQSCO można udostępnić w taki sam sposób, jak strukturę MQCD, określając adres w programie *SSLConfigPtr*lub określając przesunięcie w programie *SSLConfigOffset* . Szczegółowe informacje na ten temat zawiera opis komendy *ClientConnPtr* . Można jednak używać nie więcej niż jednego z następujących produktów: *SSLConfigPtr* i *SSLConfigOffset* ; Wywołanie nie powiodło się z kodem przyczyny MQRC_SSL_CONFIG_ERROR. jeśli obie wartości są niezerowe.

Po zakończeniu wywołania MQCONNX struktura MQSCO nie jest ponownie przywoływana.

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_4.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

SSLConfigOffset (MQLONG) dla MQCNO

SSLConfigOffset jest przesunięciem w bajtach struktury MQSCO od początku struktury MQCNO. Przesunięcie może być dodatnie lub ujemne. To pole jest polem wejściowym o wartości początkowej 0.

Parametru *SSLConfigOffset* należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client. Informacje na temat sposobu użycia tego pola zawiera opis pola *SSLConfigPtr* .

To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_4.

ConnectionId (MQBYTE24) dla MQCNO

ConnectionId to unikalny 24-bajtowy identyfikator, który umożliwia produktowi IBM MQ niezawodną identyfikację aplikacji. Aplikacja może użyć tego identyfikatora do korelacji w wywołaniach PUT i GET. Ten parametr wyjściowy ma początkową wartość 24 bajtów pustych we wszystkich językach programowania.

Menedżer kolejek przypisuje unikalny identyfikator do wszystkich połączeń, jednak są one nawiązywane. Jeśli MQCONNX nawiązuje połączenie z MQCNO w wersji 5, aplikacja może określić wartość ConnectionId ze zwróconego MQCNO. Przypisany identyfikator musi być unikalny wśród wszystkich innych identyfikatorów generowanych przez produkt IBM MQ , takich jak CorrelId, MsgIDi GroupId.

Identyfikator ConnectionId służy do identyfikowania długotrwałych jednostek pracy za pomocą komendy PCF Inquire Connection lub komendy MQSC DISPLAY CONN. Identyfikator ConnectionId używany przez komendy MQSC (CONN) pochodzi ze zwróconego tutaj identyfikatora ConnectionId . Komendy PCF Inquire i Stop Connection mogą używać zwróconego tutaj identyfikatora ConnectionId bez modyfikacji.

Aby wymusić zakończenie długotrwałego działania jednostki pracy, można użyć ConnectionId , podając ConnectionId za pomocą komendy PCF Stop Connection lub komendy MQSC STOP CONN. Więcej informacji na temat używania tych komend można znaleźć w sekcji Zatrzymywanie połączenia i w sekcji STOP CONN .

To pole nie jest zwracane, jeśli wersja jest wcześniejsza niż MQCNO_VERSION_5.

Długość tego pola jest określona przez wartość MQ_CONNECTION_ID_LENGTH.

SecurityParmsPrzesunięcie (MQLONG) dla MQCNO

SecurityParmsPrzesunięcie jest przesunięciem w bajtach struktury MQCSP od początku struktury MQCNO. Przesunięcie może być dodatnie lub ujemne. To pole jest polem wejściowym o wartości początkowej 0.

To pole jest ignorowane, jeśli wartość w polu *Wersja* jest mniejsza niż MQCNO_VERSION_5.

Struktura MQCSP jest zdefiniowana w pliku “MQCSP-parametry zabezpieczeń” na stronie 341.

SecurityParmsPtr (PMQCSP) dla MQCNO

SecurityParmsPtr to adres struktury MQCSP używany do określenia identyfikatora użytkownika i hasła na potrzeby uwierzytelniania przez usługę autoryzacji. To pole jest polem wejściowym, a jego wartością początkową jest wskaźnik pusty lub bajty puste.

To pole jest ignorowane, jeśli wartość w polu *Wersja* jest mniejsza niż MQCNO_VERSION_5.

Struktura MQCSP jest zdefiniowana w pliku “MQCSP-parametry zabezpieczeń” na stronie 341.

Zarezerwowane (MQBYTE4) dla MQCNO

Pole zastrzeżone umożliwiające dopełnienie struktury do granicy 64-bitowej. Wartość początkowa pola jest zerem binarnym dla długości pola.

To pole jest ignorowane, jeśli wartość Version jest mniejsza niż MQCNO_VERSION_6.

CCDTUrlLength (MQLONG) dla MQCNO

CCDTUrlLength to długość łańcucha identyfikowanego przez łańcuch CCDTUrlPtr lub CCDTUrlOffset , który zawiera URL identyfikujący położenie tabeli kanału połączenia klienckiego, która ma być używana dla połączenia. Wartością początkową pola jest zero.

Parametru `CCDTUrlLength` należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client.

Jest to programowa alternatywa dla ustawiania zmiennych środowiskowych MQCHLLIB i MQCHLTAB .

Jeśli aplikacja nie działa jako klient, parametr `CCDTUrlLength` jest ignorowany.

To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_6.

CCDTUrlPtr (PMQCHAR) dla MQCNO

CCDTUrlPtr to opcjonalny wskaźnik do łańcucha zawierającego URL, służący do identyfikowania położenia tabeli kanału połączenia klienckiego, która ma być używana dla połączenia. To pole jest polem wejściowym z wartością początkową wskaźnika pustego w językach programowania, które obsługują wskaźniki, i w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Parametru `CCDTUrlPtr` należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client.

Ważne: Można użyć tylko jednej z następujących wartości: `CCDTUrlPtr` i `CCDTUrlOffset`. Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CCDT_URL_ERROR, jeśli oba pola są niezerowe.

Jest to programowa alternatywa dla ustawiania zmiennych środowiskowych MQCHLLIB i MQCHLTAB .

Jeśli aplikacja nie działa jako klient, parametr `CCDTUrlPtr` jest ignorowany.

To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_6.

CCDTUrlOffset (MQLONG) dla MQCNO

CCDTUrlOffset jest przesunięciem w bajtach od początku struktury MQCNO do łańcucha zawierającego URL identyfikujący położenie tabeli kanału połączenia klienckiego, która ma być używana dla połączenia. Przesunięcie może być dodatnie lub ujemne, a wartość początkowa pola wynosi zero.

Parametru `CCDTUrlOffset` należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client.

Ważne: Można użyć tylko jednej z następujących wartości: `CCDTUrlPtr` i `CCDTUrlOffset`. Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CCDT_URL_ERROR, jeśli oba pola są niezerowe.

Jest to programowa alternatywa dla ustawiania zmiennych środowiskowych MQCHLLIB i MQCHLTAB .

Jeśli aplikacja nie działa jako klient, parametr `CCDTUrlOffset` jest ignorowany.

To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_6.

ApplName (MQCHAR28) dla MQCNO

Nazwa ustawiona przez aplikację w celu zidentyfikowania połączenia z menedżerem kolejek. Wartością początkową pola jest MQAN_NONE_ARRAY (znaki puste).

To pole jest ignorowane, jeśli `Version` jest mniejsze niż MQCNO_VERSION_7lub jeśli wartość jest pusta.

z/OS Tego pola nie można ustawić w systemie z/OS. Jeśli zostanie podjęta taka próba, zostanie zwrócony kod przyczyny MQRC_CNO_ERROR.

Reserved2 (MQBYTE4) dla MQCNO

Pole zastrzeżone umożliwiające dopełnienie struktury do granicy 64-bitowej. Wartość początkowa pola jest zerem binarnym dla długości pola.

To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_7.

V 9.3.0 *BalanceParmsPrzesunięcie (MQLONG) dla MQCNO*

Położenie pamięci dla struktury typu MQBNO, która zawiera informacje o zachowaniu aplikacji podczas równoważenia. Struktura jest całkowicie ignorowana, chyba że aplikacja łączy się za pośrednictwem kanału klienta.

To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_8.

Więcej informacji na ten temat zawiera sekcja MQBNO .

W przypadku podania tego pola nie można podać pola “BalanceParmsPtr (MQPTR) dla MQCNO” na stronie 341 . W przypadku próby podania obu pól zostanie wyświetlony komunikat MQRC_CNO_ERROR. Ponieważ to pole dotyczy tylko połączeń klienckich, podanie tego pola dla każdego innego typu połączenia powoduje również błąd MQRC_CNO_ERROR.

V 9.3.0 *BalanceParmsPtr (MQPTR) dla MQCNO*

Wskaźnik do położenia pamięci dla struktury typu MQBNO, która zawiera informacje o zachowaniu aplikacji podczas równoważenia. Struktura jest całkowicie ignorowana, chyba że aplikacja łączy się za pośrednictwem kanału klienta.

To pole jest ignorowane, jeśli wartość `Version` jest mniejsza niż MQCNO_VERSION_8.

Więcej informacji na ten temat zawiera sekcja MQBNO .

W przypadku podania tego pola nie można podać pola “BalanceParmsPrzesunięcie (MQLONG) dla MQCNO” na stronie 341 . W przypadku próby podania obu pól zostanie wyświetlony komunikat MQRC_CNO_ERROR. Ponieważ to pole dotyczy tylko połączeń klienckich, podanie tego pola dla każdego innego typu połączenia powoduje również błąd MQRC_CNO_ERROR.

MQCSP-parametry zabezpieczeń

Struktura parametrów zabezpieczeń połączenia IBM MQ jest używana przez aplikacje do przekazywania informacji uwierzytelniających w wywołaniu MQCONNX do menedżera kolejek. Można go również użyć do udostępnienia początkowego klucza używanego z systemem ochrony hasłem IBM MQ , który szyfruje poufne dane.

Ustaw parametr *`AuthenticationType`* na wartość `MQCSP_AUTH_USER_ID_AND_PWD` , aby dołączyć ID użytkownika i hasło z wersji 1.

Po podaniu początkowych informacji o kluczu w wersji 2 wartością domyślną parametru *`AuthenticationType`* jest `MQCSP_AUTH_NONE`.

V 9.3.4 W produkcie IBM MQ 9.3.4należy użyć pliku *`AuthenticationType`* , aby dołączyć informacje o znaczniku uwierzytelniania.

V 9.3.4 Można użyć opcji `MQCSP_AUTH_USER_ID_AND_PWD` lub `MQCSP_AUTH_ID_TOKEN`, ale nie obu tych opcji.

Ostrzeżenie: W niektórych przypadkach hasło lub znacznik uwierzytelniania w strukturze MQCSP dla aplikacji klienckiej jest przesyłane przez sieć w postaci jawnego tekstu. Aby upewnić się, że hasła aplikacji klienckiej i znaczniki uwierzytelniania są odpowiednio chronione, należy zapoznać się z sekcją Ochrona hasłem MQCSP.

Dostępność

Struktura MQCSP jest dostępna na wszystkich obsługiwanych platformach IBM MQ .

Wersja

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję protokołu MQCSP, ale z wartością początkową pola *`Version`* ustawioną na wartość MQCSP_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *`Version`* wymagany numer wersji.

Zestaw znaków i kodowanie

Dane w MQCSP muszą być w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one podawane odpowiednio przez atrybut menedżera kolejek **CodedCharSetId** i parametr MQENC_NATIVE.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 482. Pola w MQCSP dla MQCSP

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQCSP_STRUC_ID	'CSP¬'
Wersja (numer wersji struktury)	MQCSP_VERSION_1	1
AuthenticationType (typ uwierzytelniania)	Brak	MQCSP_AUTH_NONE
Reserved1 (wymagane do wyrównania wskaźnika w systemie IBM i)	Brak	Pusty łańcuch lub odstępy
CSPUserIdPtr (adres ID użytkownika)	Brak	Pusty wskaźnik lub puste bajty
CSPUserIdPrzesunięcie (przesunięcie identyfikatora użytkownika)	Brak	0
CSPUserIdDługość (długość identyfikatora użytkownika)	Brak	0
Reserved2 (wymagane do wyrównania wskaźnika w systemie IBM i)	Brak	Pusty łańcuch lub odstępy
CSPPasswordPtr (adres hasła)	Brak	Pusty wskaźnik lub puste bajty
CSPPasswordOffset (przesunięcie hasła)	Brak	0
CSPPasswordLength (długość hasła)	Brak	0
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_2.		
V 9.3.0 V 9.3.0 Reserved3 (wymagane do wyrównania wskaźnika w systemie IBM i)	Brak	Pusty łańcuch lub odstępy
V 9.3.0 V 9.3.0 InitialKeyPtr	Brak	Pusty wskaźnik lub odstępy
V 9.3.0 V 9.3.0 InitialKeyOffset (przesunięcie klucza początkowego dla systemu ochrony hasła)	Brak	0
V 9.3.0 V 9.3.0 InitialKeyLength (długość klucza początkowego dla systemu ochrony hasła)	Brak	0
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_3.		
V 9.3.4 Reserved4 (wymagane do wyrównania wskaźnika w systemie IBM i)	Brak	Pusty łańcuch lub odstępy

Tabela 482. Pola w MQCSP dla MQCSP (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
V 9.3.4 TokenPtr (adres znacznika uwierzytelniania)	Brak	Pusty wskaźnik lub odstępy
V 9.3.4 TokenKeyOffset (przesunięcie znacznika uwierzytelniania)	Brak	0
V 9.3.4 TokenLength (długość znacznika uwierzytelniania)	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraParametr MQCSP_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCSP MyCSP = {MQCSP_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQCSP

V 9.3.0 V 9.3.0

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
  MQLONG     AuthenticationType; /* Type of authentication */
  MQBYTE4    Reserved1;          /* Required for IBM i pointer alignment */
  MQPTR      CSPUserIdPtr;       /* Address of user ID */
  MQLONG     CSPUserIdOffset;    /* Offset of user ID */
  MQLONG     CSPUserIdLength;    /* Length of user ID */
  MQBYTE8    Reserved2;          /* Required for IBM i pointer alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
  MQLONG     CSPPasswordOffset;  /* Offset of password */
  MQLONG     CSPPasswordLength;  /* Length of password */
/* Ver:1 */

  MQBYTE8    Reserved3;          /* Required for IBM i pointer alignment */
  MQPTR      InitialKeyPtr;      /* Address of initial key */
  MQLONG     InitialKeyOffset;   /* Offset of initial key */
  MQLONG     InitialKeyLength;   /* Length of initial key */
/* Ver:2 */
  };
```

V 9.3.4

```
typedef struct tagMQCSP MQCSP;
struct tagMQCSP {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
  MQLONG     AuthenticationType; /* Type of authentication */
  MQBYTE4    Reserved1;          /* Required for IBM i pointer alignment */
  MQPTR      CSPUserIdPtr;       /* Address of user ID */
  MQLONG     CSPUserIdOffset;    /* Offset of user ID */
  MQLONG     CSPUserIdLength;    /* Length of user ID */
  MQBYTE8    Reserved2;          /* Required for IBM i pointer alignment */
  MQPTR      CSPPasswordPtr;     /* Address of password */
  MQLONG     CSPPasswordOffset;  /* Offset of password */
  MQLONG     CSPPasswordLength;  /* Length of password */
/* Ver:1 */

  MQBYTE8    Reserved3;          /* Required for IBM i pointer alignment */
  MQPTR      InitialKeyPtr;      /* Address of initial key */
```

```
  MQLONG     InitialKeyOffset;   /* Offset of initial key */
  MQLONG     InitialKeyLength;   /* Length of initial key */
/* Ver:2 */

  MQBYTE8    Reserved4;          /* Required for IBM i pointer alignment */
  MQPTR      TokenPtr;           /* Address of token */
  MQLONG     TokenOffset;        /* Offset of token */
  MQLONG     TokenLength;        /* Length of token */
/* Ver:3 */
  };
```

Deklaracja COBOL dla MQCSP

V 9.3.0 V 9.3.0

```
**   MQCSP structure
  10 MQCSP.
**    Structure identifier
   15 MQCSP-STRUCID           PIC X(4).
**    Structure version number
   15 MQCSP-VERSION           PIC S9(9) BINARY.
**    Type of authentication
   15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED1         PIC X(4).
**    Address of user ID
   15 MQCSP-CSPUSERIDPTR      POINTER.
**    Offset of user ID
   15 MQCSP-CSPUSERIDOFFSET   PIC S9(9) BINARY.
**    Length of user ID
   15 MQCSP-CSPUSERIDLENGTH   PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED2         PIC X(4).
**    Address of password
   15 MQCSP-CSPPASSWORDPTR    POINTER.
**    Offset of password
   15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**    Length of password
   15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

**    Reserved
   15 MQCSP-RESERVED3         PIC X(8).
**    Address of initial key
   15 MQCSP-INITIALKEYPTR     POINTER.
**    Offset of initial key
   15 MQCSP-INITIALKEYOFFSET  PIC S9(9) BINARY.
**    Length of initial key
   15 MQCSP-INITIALKEYLENGTH  PIC S9(9) BINARY.
** Ver:2 **
```

V 9.3.4

```
**   MQCSP structure
  10 MQCSP.
**    Structure identifier
   15 MQCSP-STRUCID           PIC X(4).
**    Structure version number
   15 MQCSP-VERSION           PIC S9(9) BINARY.
**    Type of authentication
   15 MQCSP-AUTHENTICATIONTYPE PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED1         PIC X(4).
**    Address of user ID
   15 MQCSP-CSPUSERIDPTR      POINTER.
**    Offset of user ID
   15 MQCSP-CSPUSERIDOFFSET   PIC S9(9) BINARY.
**    Length of user ID
   15 MQCSP-CSPUSERIDLENGTH   PIC S9(9) BINARY.
**    Required for IBM i pointer alignment
   15 MQCSP-RESERVED2         PIC X(4).
**    Address of password
   15 MQCSP-CSPPASSWORDPTR    POINTER.
**    Offset of password
   15 MQCSP-CSPPASSWORDOFFSET PIC S9(9) BINARY.
**    Length of password
   15 MQCSP-CSPPASSWORDLENGTH PIC S9(9) BINARY.
** Ver:1 **

**    Reserved
```

```
   15 MQCSP-RESERVED3         PIC X(8).
**    Address of initial key
   15 MQCSP-INITIALKEYPTR     POINTER.
**    Offset of initial key
   15 MQCSP-INITIALKEYOFFSET  PIC S9(9) BINARY.
**    Length of initial key
   15 MQCSP-INITIALKEYLENGTH  PIC S9(9) BINARY.
** Ver:2 **

**    Reserved
   15 MQCSP-RESERVED4         PIC X(8).
**    Address of token
   15 MQCSP-TOKENPTR     POINTER.
**    Offset of token
   15 MQCSP-TOKENOFFSET  PIC S9(9) BINARY.
**    Length of token
   15 MQCSP-TOKENLENGTH  PIC S9(9) BINARY.
** Ver:3 **
```

Deklaracja języka PL/I dla protokołu MQCSP

V 9.3.0 V 9.3.0

```
dcl
 1 MQCSP based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 AuthenticationType fixed bin(31),  /* Type of authentication */
  3 Reserved1         char(4),       /* Required for IBM i pointer
                                        alignment */
  3 CSPUserIdPtr      pointer,       /* Address of user ID */
  3 CSPUserIdOffset   fixed bin(31), /* Offset of user ID */
  3 CSPUserIdLength   fixed bin(31), /* Length of user ID */
  3 Reserved2         char(8),       /* Required for IBM i pointer
                                        alignment */
  3 CSPPasswordPtr    pointer,       /* Address of password */
  3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
  3 CSPPasswordLength fixed bin(31), /* Length of user ID */
/* Version 1 */

  3 Reserved3         char(8),       /* Reserved */
  3 InitialKeyPtr     pointer,       /* Address of initial key */
  3 InitialKeyOffset  fixed bin(31), /* Offset of initial key */
  3 InitialKeyLength  fixed bin(31); /* Length of initial key */
/* Version 2 */
```

V 9.3.4

```
dcl
 1 MQCSP based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 AuthenticationType fixed bin(31),  /* Type of authentication */
  3 Reserved1         char(4),       /* Required for IBM i pointer
                                        alignment */
  3 CSPUserIdPtr      pointer,       /* Address of user ID */
  3 CSPUserIdOffset   fixed bin(31), /* Offset of user ID */
  3 CSPUserIdLength   fixed bin(31), /* Length of user ID */
  3 Reserved2         char(8),       /* Required for IBM i pointer
                                        alignment */
  3 CSPPasswordPtr    pointer,       /* Address of password */
  3 CSPPasswordOffset fixed bin(31), /* Offset of user ID */
  3 CSPPasswordLength fixed bin(31), /* Length of user ID */
/* Version 1 */

  3 Reserved3         char(8),       /* Reserved */
  3 InitialKeyPtr     pointer,       /* Address of initial key */
  3 InitialKeyOffset  fixed bin(31), /* Offset of initial key */
  3 InitialKeyLength  fixed bin(31); /* Length of initial key */
/* Version 2 */

  3 Reserved4         char(8),       /* Reserved */
  3 TokenPtr          pointer,       /* Address of Token */
  3 TokenOffset       fixed bin(31), /* Offset of Token */
  3 TokenLength       fixed bin(31); /* Length of Token */
/* Version 3 */
```

Deklaracja Visual Basic dla MQCSP

```
Type MQCSP
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  AuthenticationType As Long      'Type of authentication'
  Reserved1         As MQBYTE4    'Required for IBM i pointer'
                                  'alignment'
  CSPUserIdPtr      As MQPTR      'Address of user ID'
  CSPUserIdOffset   As Long       'Offset of user ID'
  CSPUserIdLength   As Long       'Length of user ID'
  Reserved2         As MQBYTE8    'Required for IBM i pointer'
                                  'alignment'
  CSPPasswordPtr    As MQPTR      'Address of password'
  CSPPasswordOffset As Long       'Offset of password'
  CSPPasswordLength As Long       'Length of password'
End Type
```

Pojęcia pokrewne

Praca ze znacznikami uwierzytelniania

StrucId (MQCHAR4) dla MQCSP

Jest to identyfikator struktury struktury parametrów zabezpieczeń. Jest to zawsze pole wejściowe. Jego wartością jest MQCSP_STRUC_ID.

Wartość musi być następująca:

MQCSP_STRUC_ID
Identyfikator struktury parametrów zabezpieczeń.

Dla języka programowania C zdefiniowana jest również stała MQCSP_STRUC_ID_ARRAY. Ta wartość jest taka sama, jak wartość MQCSP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQCSP

Numer wersji struktury MQCSP.

Wartość musi być następująca:

MQCSP_VERSION_1
Struktura parametrów zabezpieczeń Version-1 . W wersji 1 można dołączyć identyfikator użytkownika i hasło do struktury MQCSP w celu uwierzytelnienia w menedżerze kolejek.

V 9.3.0 V 9.3.0 **MQCSP_VERSION_2**
Struktura parametrów zabezpieczeń Version-2 . W wersji 2 można dołączyć identyfikator użytkownika i hasło, aby uwierzytelnić się w menedżerze kolejek i określić klucz początkowy używany do ochrony haseł.

V 9.3.4 **MQCSP_VERSION_3**
Struktura parametrów zabezpieczeń Version-3 . W wersji 3 można dołączyć identyfikator użytkownika i hasło lub znacznik uwierzytelniania do struktury MQCSP w celu uwierzytelnienia w menedżerze kolejek. Można również określić klucz początkowy, który jest używany do ochrony haseł.

Następująca stała określa numer wersji bieżącej:

MQCSP_CURRENT_VERSION
Bieżąca wersja struktury parametrów zabezpieczeń.

Jest to zawsze pole wejściowe. Początkowa wartość tego pola to MQCSP_VERSION_3.

AuthenticationType (MQLONG) dla protokołu MQCSP

AuthenticationType jest polem wejściowym. Jego wartością początkową jest MQCSP_AUTH_NONE.

Jest to typ wykonywanego uwierzytelniania. Poprawne wartości:

MQCSP_AUTH_BRAK
Nie należy używać pól identyfikatora użytkownika i hasła ani znacznika uwierzytelniania .

MQCSP_AUTH_ID_UŻYTKOWNIKA AND_PWD
Uwierzytelnianie przy użyciu identyfikatora użytkownika i hasła w strukturze MQCSP.

V 9.3.4 **MQCSP_AUTH_ID_TOKEN**
Uwierzytelnianie przy użyciu znacznika uwierzytelniania w strukturze MQCSP.

Wartością domyślną jest MQCSP_AUTH_NONE. W przypadku ustawienia domyślnego zabezpieczenia hasłem nie są wykonywane.

Jeśli wymagane jest uwierzytelnianie, należy ustawić wartość **MQCSP**.**AuthenticationType** na MQCSP_AUTH_USER_ID_AND_PWD lub MQCSP_AUTH_ID_TOKEN.

Więcej informacji na ten temat zawiera sekcja Ochrona hasłem MQCSP.

Pojęcia pokrewne
Praca ze znacznikami uwierzytelniania

Reserved1 (MQBYTE4) dla MQCSP
Pole zastrzeżone, wymagane do wyrównania wskaźnika w systemie IBM i.

Początkowa wartość tego pola jest pusta.

CSPUserIdPtr (MQPTR) dla MQCSP
Adres dla ID użytkownika, który ma być używany podczas uwierzytelniania.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_5.

To pole może zawierać identyfikator użytkownika systemu operacyjnego, jeśli w polu CONNAUTH menedżera kolejek znajduje się nazwa **AUTHTYPE** lub *IDPWOS* .

W systemie Windows może to być pełny identyfikator użytkownika domeny.

To pole może zawierać identyfikator użytkownika LDAP, jeśli w polu CONNAUTH menedżera kolejek podano nazwę **AUTHTYPE** lub *IDPWLDAP* .

CSPUserIdPrzesunięcie (MQLONG) dla MQCSP
Przesunięcie w bajtach dla ID użytkownika, który ma być używany w uwierzytelnianiu. Przesunięcie może być dodatnie lub ujemne.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

CSPUserIdDługość (MQLONG) dla MQCSP
Długość identyfikatora użytkownika, który ma być używany podczas uwierzytelniania.

Maksymalna długość identyfikatora użytkownika zależy od platformy (patrz sekcja Identyfikatory użytkowników). V 9.3.0 V 9.3.0 Jeśli długość identyfikatora użytkownika przekracza maksymalną dozwoloną długość, żądanie uwierzytelniania kończy się niepowodzeniem z błędem MQRC_CSP_ERROR. We wcześniejszych wersjach produktu IBM MQzwracany jest błąd MQRC_NOT_AUTHORIZED.

To pole jest polem wejściowym. Wartością początkową tego pola jest 0.

Reserved2 (MQBYTE8) dla MQCSP
Pole zastrzeżone, wymagane do wyrównania wskaźnika w systemie IBM i.

Początkowa wartość tego pola jest pusta.

CSPPasswordPtr (MQPTR) dla MQCSP
Adres hasła, które ma być używane w uwierzytelnianiu.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCNO_VERSION_5.

To pole może zawierać puste hasło, które jest odrzucane przez system operacyjny lub sprawdzanie hasła LDAP, w zależności od konfiguracji, ale nie jest odrzucane przez IBM MQ przed przekazaniem go do metody uwierzytelniania.

CSPPasswordOffset (MQLONG) dla MQCSP

Jest to przesunięcie w bajtach dla hasła, które ma być używane w uwierzytelnianiu. Przesunięcie może być dodatnie lub ujemne.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

CSPPasswordLength (MQLONG) dla MQCSP

Długość hasła, które ma być używane w uwierzytelnianiu.

Maksymalna długość hasła to MQ_CSP_PASSWORD_LENGTH, która wynosi 256 znaków.

V 9.3.0 V 9.3.0 Jeśli długość hasła przekracza maksymalną dozwoloną długość, żądanie uwierzytelniania kończy się niepowodzeniem z błędem MQRC_CSP_ERROR. We wcześniejszych wersjach produktu IBM MQzwracany jest błąd MQRC_NOT_AUTHORIZED.

To pole jest polem wejściowym. Wartością początkową tego pola jest 0.

V 9.3.0 V 9.3.0 *Reserved3 (MQBYTE8) dla MQCSP*

Pole zastrzeżone, wymagane do wyrównania wskaźnika w systemie IBM i.

Początkowa wartość tego pola jest pusta.

V 9.3.0 Multi V 9.3.0 *InitialKeyPtr (MQPTR) dla MQCSP*

Adres początkowego klucza systemu zabezpieczenia hasłem.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_2.

To pole dotyczy tylko systemu IBM MQ MQI clients działającego w systemach IBM ii AIX, Linux, and Windows .

IBM MQ MQI clients może podać zaszyfrowane wartości w niektórych polach, używając systemu ochrony hasłem IBM MQ . Jeśli do zaszyfrowania hasła repozytorium kluczy określonego w strukturze MQCSO użyto klucza początkowego, należy upewnić się, że w protokole MQCSP dla tej samej aplikacji klienckiej zostały uwzględnione pola klucza początkowego.

Klienty MQI produktu IBM MQ mogą podawać wartości szyfrowane w niektórych polach za pomocą systemu zabezpieczenia hasłem produktu IBM MQ . Jeśli do zaszyfrowania hasła repozytorium kluczy określonego w strukturze MQCSO użyto klucza początkowego, należy upewnić się, że w strukturze MQCSP dla tej samej aplikacji klienckiej zostały uwzględnione pola klucza początkowego.

Klucz początkowy jest używany przez algorytm szyfrowania do szyfrowania i deszyfrowania tych wartości. Jeśli klucz początkowy zostanie podany, gdy wartości tych pól są szyfrowane za pomocą programu narzędziowego **runmqicred** , klient musi określić ten sam klucz początkowy podczas nawiązywania połączenia z menedżerem kolejek.

Klucz początkowy określony za pomocą tego pola nadpisuje klucz początkowy określony za pomocą zmiennej środowiskowej *MQS_MQI_KEYFILE* lub właściwości *MQIInitialKeyFile* w sekcji Security pliku konfiguracyjnego klienta.

Do określenia klucza początkowego można użyć wartości *InitialKeyOffset* lub *InitialKeyPtr* , ale nie obu tych wartości jednocześnie.

Zadania pokrewne

Podawanie klucza początkowego dla klienta MQI produktu IBM MQ w systemach AIX, Linux i Windows

Ochrona haseł w plikach konfiguracyjnych komponentu IBM MQ

Odsyłacze pokrewne

runmqicred (chroń hasła klienta IBM MQ)

“KeyRepoPasswordPtr (MQPTR) dla MQSCO” na stronie 583

Jest to adres w bajtach hasła repozytorium kluczy TLS.

“Przesunięcie InitialKey(MQLONG) dla MQCSP” na stronie 349

Przesunięcie w bajtach dla klucza początkowego systemu zabezpieczenia hasłem od początku struktury MQCSP. Przesunięcie może być dodatnie lub ujemne.

V 9.3.0 Multi V 9.3.0 *Przesunięcie InitialKey(MQLONG) dla MQCSP*

Przesunięcie w bajtach dla klucza początkowego systemu zabezpieczenia hasłem od początku struktury MQCSP. Przesunięcie może być dodatnie lub ujemne.

Do określenia klucza początkowego można użyć wartości *InitialKeyOffset* lub *InitialKeyPtr* , ale nie obu tych wartości jednocześnie. Więcej informacji na ten temat zawiera opis pola *InitialKeyPtr* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_2.

Zadania pokrewne

Ochrona haseł w plikach konfiguracyjnych komponentu IBM MQ

Podawanie klucza początkowego dla klienta MQI produktu IBM MQ w systemach AIX, Linux i Windows

Odsyłacze pokrewne

“InitialKeyPtr (MQPTR) dla MQCSP” na stronie 348

Adres początkowego klucza systemu zabezpieczenia hasłem.

V 9.3.0 Multi V 9.3.0 *InitialKeyDługość (MQLONG) dla MQCSP*

Długość klucza początkowego dla systemu zabezpieczenia hasłem.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_2.

Zadania pokrewne

Ochrona haseł w plikach konfiguracyjnych komponentu IBM MQ

Podawanie klucza początkowego dla klienta MQI produktu IBM MQ w systemach AIX, Linux i Windows

V 9.3.4 *Reserved4 (MQBYTE8) dla MQCSP*

Pole zastrzeżone, wymagane do wyrównania wskaźnika w systemie IBM i.

Początkowa wartość tego pola jest pusta.

V 9.3.4 *TokenPtr (MQPTR) dla MQCSP*

Adres znacznika uwierzytelniania używanego do uwierzytelniania w menedżerze kolejek.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_3.

To pole dotyczy IBM MQ MQI clients nawiązywania połączeń z menedżerami kolejek systemu IBM MQ , które działają w systemach AIX lub Linux .

Do określenia znacznika uwierzytelniania można użyć jednej z następujących metod: *TokenOffset* lub *TokenPtr* , ale nie obu.

Więcej informacji na ten temat zawiera sekcja Używanie znaczników uwierzytelniania w aplikacji.

Pojęcia pokrewne

Praca ze znacznikami uwierzytelniania

Odsyłacze pokrewne

“TokenOffset (MQLONG) dla MQCSP” na stronie 350

Jest to przesunięcie w bajtach dla znacznika uwierzytelniania od początku struktury MQCSP. Przesunięcie może być dodatnie lub ujemne.

V 9.3.4 *TokenOffset (MQLONG) dla MQCSP*

Jest to przesunięcie w bajtach dla znacznika uwierzytelniania od początku struktury MQCSP. Przesunięcie może być dodatnie lub ujemne.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_3.

Do określenia znacznika można użyć parametru *TokenOffset* lub *TokenPtr* , ale nie obu tych parametrów. Więcej informacji na ten temat zawiera opis pola *TokenPtr* .

Pojęcia pokrewne

Praca ze znacznikami uwierzytelniania

Zadania pokrewne

Używanie znaczników uwierzytelniania w aplikacji

Odsyłacze pokrewne

“TokenPtr (MQPTR) dla MQCSP” na stronie 349

Adres znacznika uwierzytelniania używanego do uwierzytelniania w menedżerze kolejek.

V 9.3.4 *TokenLength (MQLONG) dla MQCSP*

Jest to długość znacznika uwierzytelniania używanego do uwierzytelniania w menedżerze kolejek.

Maksymalna długość znacznika uwierzytelniania to MQ_CSP_TOKEN_LENGTH, która wynosi 8192 bajtów. Jeśli wartość parametru *TokenLength* jest większa niż maksymalna dozwolona długość, żądanie uwierzytelniania kończy się niepowodzeniem z błędem MQRC_CSP_ERROR.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQCSP_VERSION_3.

Więcej informacji na ten temat zawiera sekcja Używanie znaczników uwierzytelniania w aplikacji.

Pojęcia pokrewne

Praca ze znacznikami uwierzytelniania

MQCTLO-struktura opcji wywołania zwrotnego sterowania

Struktura MQCTLO służy do określania opcji związanych z funkcją zwrotną sterowania. Struktura jest parametrem wejściowym i wyjściowym wywołania MQCTL.

Dostępność

Struktura MQCTLO jest dostępna na następujących platformach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows
- z/OS z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Wersja

Bieżąca wersja obiektu MQCTLO to MQCTLO_VERSION_1.

Zestaw znaków i kodowanie

Dane w obiekcie MQCTLO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 483. Pola w MQCTLO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucID (identyfikator struktury)	Identyfikator struktury MQCTLO_STRUC_ID	'CTLO'
Wersja (numer wersji struktury)	MQCTLO_VERSION_1	1
Opcje (opcje)	MQCTLO_NONE	Wartości null
Opcje (pole zastrzeżone)	Pole zastrzeżone	
ConnectionArea (pole używane przez funkcję zwrotną)	Brak	Pusty wskaźnik lub puste bajty

Uwagi:

1. W języku programowania C: zmienna makraMQCTLO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQCTLO MyCTLO = {MQCTLO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQCTLO

```
typedef struct tagMQCTLO MQCTLO;
 struct tagMQCTLO {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    Options;        /* Options that control the action of MQCTL */
  MQLONG    Reserved;       /* Reserved field */


MQPTR     ConnectionArea; /* Connection work area passed to the function */
 };
```

Deklaracja języka COBOL dla MQCTLO

```
** MQCTLO structure
 10  MQCTLO.
** Structure Identifier
 15  MQCTLO-STRUCID                    PIC X(4).
** Structure Version
 15  MQCTLO-VERSION                    PIC S9(9) BINARY.
 ** Options
```

```
 15  MQCTLO-OPTIONS                      PIC S9(9) BINARY.
** Reserved
 15  MQCTLO-RESERVED                     PIC S9(9) BINARY.
** ConnectionArea
 15  MQCTLO-CONNECTIONAREA               POINTER
```

Deklaracja PL/I dla MQCTLO

```
dcl
 1 MQCTLO based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version */
  3 Options             fixed bin(31), /* Options */
  3 Reserved            fixed bin(31),
  3 ConnectionArea      pointer;       /* Connection work area */
```

StrucId (MQCHAR4) dla MQCTLO

Jest to identyfikator struktury struktury opcji sterujących. Jest to zawsze pole wejściowe. Jego wartością jest MQCTLO_STRUC_ID.

Wartość musi być następująca:

Identyfikator struktury MQCTLO_STRUC_ID
Identyfikator struktury opcji sterujących.

Dla języka programowania C zdefiniowana jest również stała MQCTLO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQCTLO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQCTLO

Struktura opcji sterujących-pole Wersja

Jest to numer wersji struktury; wartość musi być następująca:

MQCTLO_VERSION_1
Version-1 -struktura opcji sterujących.

Następująca stała określa numer wersji bieżącej:

MQCTLO_CURRENT_VERSION
Bieżąca wersja struktury opcji sterujących.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQCTLO_VERSION_1.

Opcje (MQLONG) dla MQCTLO

Struktura opcji sterujących-pole Opcje

Opcje sterujące działaniem obiektu MQCTL.

MQCTLO_FAIL_IF_QUIESCING,
Wymuś niepowodzenie wywołania MQCTL, jeśli menedżer kolejek lub połączenie jest w stanie wyciszania.

Określ opcję MQGMO_FAIL_IF_QUIESCING w opcjach MQGMO przekazywanych w wywołaniu MQCB, aby spowodować wysłanie powiadomienia do konsumentów komunikatów podczas wyciszania.

MQCTLO_THREAD_AFFINITY
Ta opcja informuje system, że aplikacja wymaga, aby wszystkie konsumenty komunikatów dla tego samego połączenia były wywoływane w tym samym wątku. Ten wątek będzie używany dla wszystkich wywołań konsumentów do momentu zatrzymania połączenia.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

MQCTLO_NONE
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. Parametr MQCTLO_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Opcja ta nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Jest to pole wejściowe. Wartością początkową pola *Options* jest MQCTLO_NONE.

Zarezerwowany (MQLONG) dla MQCTLO

Jest to pole zastrzeżone. Wartość musi być równa zero.

ConnectionArea (MQPTR) dla MQCTLO

Struktura opcji sterujących-pole ConnectionArea

Jest to pole dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest ono przekazywane bez zmian do pola ConnectionArea w strukturze MQCBC, która jest parametrem wejściowym wywołania zwrotnego.

To pole jest ignorowane w przypadku wszystkich operacji innych niż MQOP_START i MQOP_START_WAIT.

Jest to pole wejściowe i wyjściowe funkcji zwrotnej. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

MQDH-nagłówek dystrybucji

Struktura MQDH opisuje dodatkowe dane, które są obecne w komunikacie, gdy komunikat jest komunikatem listy dystrybucyjnej przechowywanym w kolejce transmisji. Komunikat listy dystrybucyjnej to komunikat wysyłany do wielu kolejek docelowych. Dodatkowe dane składają się ze struktury MQDH, po której następuje tablica rekordów MQOR i tablica rekordów MQPMR.Ta struktura jest używana przez wyspecjalizowane aplikacje, które umieszczają komunikaty bezpośrednio w kolejkach transmisji lub usuwają komunikaty z kolejek transmisji (na przykład agenty kanału komunikatów). Aplikacje, które chcą umieszczać komunikaty na listach dystrybucyjnych, nie mogą używać tej struktury. Zamiast tego muszą używać struktury MQOD w celu zdefiniowania miejsc docelowych na liście dystrybucyjnej, a struktury MQPMO w celu określenia właściwości komunikatu lub odebrania informacji o komunikatach wysyłanych do poszczególnych miejsc docelowych.

Dostępność

Struktura MQDH jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

Nazwa formatu

Usługa MQFMT_DIST_HEADER

Zestaw znaków i kodowanie

Dane w produkcie MQDH muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE.

Ustaw zestaw znaków i kodowanie MQDH w polach *CodedCharSetId* i *Encoding* w następujących polach:

- MQMD (jeśli struktura MQDH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQDH (wszystkie inne przypadki).

Użycie

Gdy aplikacja umieszcza komunikat na liście dystrybucyjnej, a niektóre lub wszystkie miejsca docelowe są zdalne, menedżer kolejek dodaje do danych komunikatu aplikacji struktury MQXQH i MQDH i umieszcza komunikat w odpowiedniej kolejce transmisji. Dlatego dane występują w następującej kolejności, gdy komunikat znajduje się w kolejce transmisji:

- Struktura MQXQH
- Struktura MQDH plus tablice rekordów MQOR i MQPMR
- Dane komunikatu aplikacji

W zależności od miejsc docelowych menedżer kolejek może wygenerować więcej niż jeden taki komunikat i umieścić go w różnych kolejkach transmisji. W tym przypadku struktury MQDH w tych komunikatach identyfikują różne podzbiory miejsc docelowych zdefiniowanych przez listę dystrybucyjną otwartą przez aplikację.

Aplikacja, która umieszcza komunikat listy dystrybucyjnej bezpośrednio w kolejce transmisji, musi być zgodna z wcześniej opisaną sekwencją i musi zapewnić, że struktura MQDH jest poprawna. Jeśli struktura MQDH nie jest poprawna, menedżer kolejek może zakończyć niepowodzeniem wywołanie MQPUT lub MQPUT1 z kodem przyczyny MQRC_DH_ERROR.

Komunikaty można przechowywać w kolejce w formie listy dystrybucyjnej tylko wtedy, gdy kolejka została zdefiniowana jako zdolna do obsługi komunikatów listy dystrybucyjnej. Patrz atrybut kolejki **DistLists** opisany w sekcji “Atrybuty kolejek” na stronie 863. Jeśli aplikacja umieszcza komunikat listy dystrybucyjnej bezpośrednio w kolejce, która nie obsługuje list dystrybucyjnych, menedżer kolejek dzieli komunikat listy dystrybucyjnej na pojedyncze komunikaty i umieszcza je w kolejce.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 484. Pola w MQDH dla MQDH

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	ID struktury MQDH_ID	'DH¬¬'
Wersja (numer wersji struktury)	MQDH_VERSION_1	1
StrucLength (długość struktury MQDH plus następujące rekordy)	Brak	0
Kodowanie (kodowanie liczbowe danych następujących po tablicy rekordów MQPMR)	Brak	0
CodedCharSetId (identyfikator zestawu znaków danych po tablicy rekordów MQPMR)	MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0
Format (nazwa formatu danych po tablicy rekordów MQPMR)	MQFMT_BRAK	Puste
Flagi (flagi ogólne)	MQDHF_BRAK	0
PutMsgRecFields (flagi wskazujące, które pola MQPMR są obecne)	MQPMRF_BRAK	0
RecsPresent (liczba rekordów obiektów)	Brak	0
ObjectRecPrzesunięcie (przesunięcie pierwszego rekordu obiektu od początku MQDH)	Brak	0

Tabela 484. Pola w MQDH dla MQDH (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
PutMsgRecOffset (przesunięcie pierwszego rekordu put-message od początku MQDH)	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQDH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQDH MyDH = {MQDH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQDH

```
typedef struct tagMQDH MQDH;
struct tagMQDH {
  MQCHAR4  StrucId;          /* Structure identifier */
  MQLONG   Version;          /* Structure version number */
  MQLONG   StrucLength;      /* Length of MQDH structure plus following
                                MQOR and MQPMR records */
  MQLONG   Encoding;         /* Numeric encoding of data that follows
                                the MQOR and MQPMR records */
  MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows the MQOR and MQPMR records */
  MQCHAR8  Format;           /* Format name of data that follows the
                                MQOR and MQPMR records */
  MQLONG   Flags;            /* General flags */
  MQLONG   PutMsgRecFields;  /* Flags indicating which MQPMR fields are
                                present */
  MQLONG   RecsPresent;      /* Number of MQOR records present */
  MQLONG   ObjectRecOffset;  /* Offset of first MQOR record from start
                                of MQDH */
  MQLONG   PutMsgRecOffset;  /* Offset of first MQPMR record from start
                                of MQDH */
};
```

Deklaracja COBOL dla MQDH

```
**   MQDH structure
  10 MQDH.
**    Structure identifier
   15 MQDH-STRUCID         PIC X(4).
**    Structure version number
   15 MQDH-VERSION         PIC S9(9) BINARY.
**    Length of MQDH structure plus following MQOR and MQPMR records
   15 MQDH-STRUCLENGTH     PIC S9(9) BINARY.
**    Numeric encoding of data that follows the MQOR and MQPMR records
   15 MQDH-ENCODING        PIC S9(9) BINARY.
**    Character set identifier of data that follows the MQOR and MQPMR
**    records
   15 MQDH-CODEDCHARSETID  PIC S9(9) BINARY.
**    Format name of data that follows the MQOR and MQPMR records
   15 MQDH-FORMAT          PIC X(8).
**    General flags
   15 MQDH-FLAGS           PIC S9(9) BINARY.
**    Flags indicating which MQPMR fields are present
   15 MQDH-PUTMSGRECFIELDS PIC S9(9) BINARY.
**    Number of MQOR records present
   15 MQDH-RECSPRESENT     PIC S9(9) BINARY.
**    Offset of first MQOR record from start of MQDH
   15 MQDH-OBJECTRECOFFSET PIC S9(9) BINARY.
**    Offset of first MQPMR record from start of MQDH
   15 MQDH-PUTMSGRECOFFSET PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQDH

```
dcl
 1 MQDH based,
  3 StrucId         char(4),       /* Structure identifier */
  3 Version         fixed bin(31), /* Structure version number */
  3 StrucLength     fixed bin(31), /* Length of MQDH structure plus
                                      following MQOR and MQPMR
                                      records */
  3 Encoding        fixed bin(31), /* Numeric encoding of data that
                                      follows the MQOR and MQPMR
                                      records */
  3 CodedCharSetId  fixed bin(31), /* Character set identifier of data
                                      that follows the MQOR and MQPMR
                                      records */
  3 Format          char(8),       /* Format name of data that follows
                                      the MQOR and MQPMR records */
  3 Flags           fixed bin(31), /* General flags */
  3 PutMsgRecFields fixed bin(31), /* Flags indicating which MQPMR
                                      fields are present */
  3 RecsPresent     fixed bin(31), /* Number of MQOR records present */
  3 ObjectRecOffset fixed bin(31), /* Offset of first MQOR record from
                                      start of MQDH */
  3 PutMsgRecOffset fixed bin(31); /* Offset of first MQPMR record from
                                      start of MQDH */
```

Deklaracja Visual Basic dla MQDH

```
Type MQDH
  StrucId         As String*4 'Structure identifier'
  Version         As Long     'Structure version number'
  StrucLength     As Long     'Length of MQDH structure plus following'
                              'MQOR and MQPMR records'
  Encoding        As Long     'Numeric encoding of data that follows'
                              'the MQOR and MQPMR records'
  CodedCharSetId  As Long     'Character set identifier of data that'
                              'follows the MQOR and MQPMR records'
  Format          As String*8 'Format name of data that follows the'
                              'MQOR and MQPMR records'
  Flags           As Long     'General flags'
  PutMsgRecFields As Long     'Flags indicating which MQPMR fields are'
                              'present'
  RecsPresent     As Long     'Number of MQOR records present'
  ObjectRecOffset As Long     'Offset of first MQOR record from start'
                              'of MQDH'
  PutMsgRecOffset As Long     'Offset of first MQPMR record from start'
                              'of MQDH'
End Type
```

StrucId (MQCHAR4) dla MQDH

Jest to identyfikator struktury struktury nagłówka dystrybucji. Jest to zawsze pole wejściowe. Jego wartością jest MQDH_STRUC_ID.

Wartość musi być następująca:

ID struktury MQDH_ID
Identyfikator struktury nagłówka dystrybucji.

Dla języka programowania C zdefiniowana jest również stała MQDH_STRUC_ID_ARRAY. Ta wartość jest taka sama jak wartość MQDH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQDH

Wartość musi być następująca:

MQDH_VERSION_1
Numer wersji struktury nagłówka dystrybucji.

Następująca stała określa numer wersji bieżącej:

MQDH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka dystrybucji.

Wartością początkową tego pola jest MQDH_VERSION_1.

StrucLength (MQLONG) dla MQDH

Jest to liczba bajtów od początku struktury MQDH do początku danych komunikatu po tablicach rekordów MQOR i MQPMR. Dane występują w następującej kolejności:

- Struktura MQDH
- Tablica rekordów MQOR
- Tablica rekordów MQPMR
- Dane komunikatu

Tablice rekordów MQOR i MQPMR są adresowane za pomocą przesunięć zawartych w strukturze MQDH. Jeśli te przesunięcia powodują powstanie nieużywanych bajtów między jedną lub większą liczbą struktur MQDH, tablicami rekordów i danymi komunikatu, nieużywane bajty muszą zostać uwzględnione w wartości *StrucLength*, ale treść tych bajtów nie jest zachowywana przez menedżer kolejek. Tablica rekordów MQPMR może występować przed tablicą rekordów MQOR.

Wartością początkową tego pola jest 0.

Kodowanie (MQLONG) dla MQDH

Jest to kodowanie liczbowe danych następujących po tablicach rekordów MQOR i MQPMR. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

CodedCharSetId (MQLONG) dla MQDH

Jest to identyfikator zestawu znaków danych następujących po tablicach rekordów MQOR i MQPMR. Nie ma on zastosowania do danych znakowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie występuje błąd, wywołanie MQGET nie zwraca wartości MQCCSI_INHERIT.

Pola MQCCSI_INHERIT nie można używać, jeśli wartością pola *PutApplType* w deskryptorze MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQDH

Jest to nazwa formatu danych następujących po tablicach rekordów MQOD i MQPMR (w zależności od tego, co wystąpi jako ostatnie).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQDH

Można podać następującą opcję:

MQDHF_NEW_MSG_IDS
Wygeneruj nowy identyfikator komunikatu dla każdego miejsca docelowego na liście dystrybucyjnej. Tę opcję należy ustawić tylko wtedy, gdy nie ma rekordów put-message lub gdy rekordy są obecne, ale nie zawierają pola *MsgId* .

Użycie tej flagi odracza generowanie identyfikatorów komunikatów do momentu, w którym komunikat listy dystrybucyjnej zostanie ostatecznie podzielony na pojedyncze komunikaty. Minimalizuje to ilość informacji sterujących, które muszą być przesyłane z komunikatem listy dystrybucyjnej.

Gdy aplikacja umieszcza komunikat na liście dystrybucyjnej, menedżer kolejek ustawia parametr MQDHF_NEW_MSG_IDS w produkcie MQDH, który jest generowany, gdy spełnione są oba poniższe warunki:

- Aplikacja nie udostępnia rekordów komunikatów wstawianych lub podane rekordy nie zawierają pola *MsgId* .
- Pole *MsgId* w strukturze MQMD ma wartość MQMI_NONE lub pole *Options* w strukturze MQPMO zawiera wartość MQPMO_NEW_MSG_ID.

Jeśli nie są potrzebne żadne opcje, należy podać następujące informacje:

MQDHF_BRAK
Nie określono żadnych opcji. Parametr MQDHF_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta stała była używana z żadną inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Wartością początkową tego pola jest MQDHF_NONE.

PutMsgRecFields (MQLONG) dla MQDH

Można podać co najmniej jedną z następujących opcji:

MQPMRF_ID_komunikatu
Pole identyfikatora komunikatu jest obecne.

MQPMRF_CORREL_ID (Identyfikator relacji MQ)
Pole identyfikatora korelacji jest obecne.

MQPMRF_GROUP_ID (identyfikator grupy MQPMRF_ID)
Pole identyfikatora grupy jest obecne.

MQPMRF_FEEDBACK
Pole informacji zwrotnej jest obecne.

MQPMRF_ACCOUNTING_TOKEN,
Rozliczanie-pole tokenu jest obecne.

Jeśli nie ma pól MQPMR, podaj następujące informacje:

MQPMRF_BRAK
Brak pól rekordu komunikatu umieszczania (put-message). Parametr MQPMRF_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta stała była używana z żadną inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Wartością początkową tego pola jest MQPMRF_NONE.

RecsPresent (MQLONG) dla MQDH

Jest to liczba miejsc docelowych. Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *RecsPresent* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

ObjectRecPrzesunięcie (MQLONG) dla MQDH

Daje to przesunięcie w bajtach pierwszego rekordu w tablicy rekordów obiektów MQOR zawierających nazwy kolejek docelowych. W tej tablicy znajdują się rekordy *RecsPresent* . Te rekordy (plus wszystkie bajty pominięte między pierwszym rekordem obiektu a poprzednim polem) są uwzględniane w długości określonej przez pole *StrucLength* .

Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *ObjectRecOffset* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

PutMsgRecOffset (MQLONG) dla MQDH

Daje to przesunięcie w bajtach pierwszego rekordu w tablicy rekordów komunikatów umieszczania MQPMR zawierających właściwości komunikatu. Jeśli istnieje, w tej tablicy znajdują się rekordy *RecsPresent* . Te rekordy (plus wszystkie bajty pominięte między pierwszym rekordem komunikatu wstawianego a poprzednim polem) są uwzględniane w długości określonej przez pole *StrucLength* .

Rekordy umieszczania komunikatów są opcjonalne. Jeśli nie podano żadnych rekordów, parametr *PutMsgRecOffset* ma wartość zero, a parametr *PutMsgRecFields* ma wartość MQPMRF_NONE.

Wartością początkową tego pola jest 0.

MQDLH-nagłówek niedostarczonego komunikatu

Struktura MQDLH opisuje informacje, które są przedrostkiem danych komunikatu aplikacji dla komunikatów w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Komunikat może pojawić się w kolejce niedostarczonych komunikatów, ponieważ menedżer kolejek lub agent kanału komunikatów przekierował go do kolejki lub aplikacja umieściła komunikat bezpośrednio w kolejce.

Nazwa formatu

MQFMT_DEAD_LETTER_HEADER

Zestaw znaków i kodowanie

Pola w strukturze MQDLH mają zestaw znaków i kodowanie określone w polach *CodedCharSetId* i *Encoding* . Są one określane w strukturze nagłówka poprzedzającej MQDLH lub w strukturze MQMD, jeśli MQDLH jest na początku danych komunikatu aplikacji.

Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Jeśli używane są klasy IBM MQ classes for Java/JMS, a strona kodowa zdefiniowana w strukturze MQMD nie jest obsługiwana przez maszynę wirtualną Java , to struktura MQDLH jest zapisywana w zestawie znaków UTF-8 .

Użycie

Aplikacje, które umieszczają komunikaty bezpośrednio w kolejce niedostarczonych komunikatów, muszą poprzedzać dane komunikatu strukturą MQDLH i inicjować pola odpowiednimi wartościami. Jednak menedżer kolejek nie wymaga obecności struktury MQDLH lub podania poprawnych wartości w polach.

Jeśli komunikat jest zbyt długi, aby umieścić go w kolejce niedostarczonych komunikatów, aplikacja musi wykonać jedną z następujących czynności:

- Obetnij dane komunikatu, aby zmieściły się w kolejce niedostarczonych komunikatów.

- Zapisz komunikat w pamięci dyskowej i umieść komunikat raportu o wyjątku w kolejce niedostarczonych komunikatów.
- Odrzuć komunikat i zwróć błąd do jego twórcy. Jeśli komunikat jest (lub może być) komunikatem krytycznym, należy to zrobić tylko wtedy, gdy wiadomo, że nadawca nadal ma kopię komunikatu, na przykład komunikat odebrany przez agenta kanału komunikatów z kanału komunikacyjnego.

To, które z powyższych działań jest odpowiednie (jeśli istnieją), zależy od projektu aplikacji.

Menedżer kolejek wykonuje przetwarzanie specjalne, gdy komunikat, który jest segmentem, jest umieszczany ze strukturą MQDLH na początku. Szczegółowe informacje można znaleźć w opisie struktury MQMDE.

Umieszczanie komunikatów w kolejce niedostarczonych komunikatów

Jeśli komunikat jest umieszczany w kolejce niedostarczonych komunikatów, struktura MQMD używana dla wywołania MQPUT lub MQPUT1 musi być taka sama jak struktura MQMD powiązana z komunikatem (zwykle jest to struktura MQMD zwracana przez wywołanie MQGET), z wyjątkiem następujących sytuacji:

- Ustaw pola *CodedCharSetId* i *Encoding* na dowolny zestaw znaków i kodowanie używane dla pól w strukturze MQDLH.
- Ustaw w polu *Format* wartość MQFMT_DEAD_LETTER_HEADER, aby wskazać, że dane rozpoczynają się od struktury MQDLH.
- Ustaw pola kontekstu (*AccountingToken*, *ApplIdentityData*, *ApplOriginData*, *PutApplName*, *PutApplType*, *PutDate*, *PutTime*, *UserIdentifier*) przy użyciu opcji kontekstu odpowiedniej dla okoliczności:
  - Aplikacja umieszczająca w kolejce niedostarczonych komunikatów komunikat, który nie jest powiązany z żadnym poprzednim komunikatem, musi korzystać z opcji MQPMO_DEFAULT_CONTEXT. Powoduje to, że menedżer kolejek ustawia wszystkie pola kontekstu w deskryptorze komunikatu na ich wartości domyślne.
  - Aplikacja serwera umieszczająca w kolejce niedostarczonych komunikatów komunikat, który właśnie otrzymała, musi użyć opcji MQPMO_PASS_ALL_CONTEXT, aby zachować oryginalne informacje o kontekście.
  - Aplikacja serwera umieszczająca w kolejce niedostarczonych komunikatów *odpowiedź* na odebrany komunikat musi korzystać z opcji MQPMO_PASS_IDENTITY_CONTEXT. Ta opcja zachowuje informacje o tożsamości, ale ustawia informacje o pochodzeniu na informacje aplikacji serwera.
  - Agent kanału komunikatów umieszczający w kolejce niedostarczonych komunikatów komunikat odebrany z kanału komunikacyjnego musi użyć opcji MQPMO_SET_ALL_CONTEXT, aby zachować oryginalne informacje o kontekście.

W samej strukturze MQDLH ustaw pola w następujący sposób:

- W polach *CodedCharSetId*, *Encoding*i *Format* ustaw wartości opisujące dane zgodne ze strukturą MQDLH, zwykle wartości z oryginalnego deskryptora komunikatu.
- W polach kontekstu *PutApplType*, *PutApplName*, *PutDate*i *PutTime* ustaw wartości odpowiednie dla aplikacji, która umieszcza komunikat w kolejce niedostarczonych komunikatów. Te wartości nie są powiązane z oryginalnym komunikatem.
- Ustaw odpowiednio inne pola.

Upewnij się, że wszystkie pola mają poprawne wartości oraz że pola znakowe są dopełniane spacjami do zdefiniowanej długości pola. Nie należy przedwcześnie kończyć danych znakowych przy użyciu znaku o kodzie zero, ponieważ menedżer kolejek nie przekształca znaków o kodzie zero i kolejnych znaków w odstępy w strukturze MQDLH.

Pobieranie komunikatów z kolejki niedostarczonych komunikatów

Aplikacje, które pobierają komunikaty z kolejki niedostarczonych komunikatów, muszą sprawdzić, czy komunikaty zaczynają się od struktury MQDLH. Aplikacja może określić, czy struktura MQDLH

jest obecna, sprawdzając pole *Format* w deskryptorze komunikatu MQMD. Jeśli pole ma wartość MQFMT_DEAD_LETTER_HEADER, dane komunikatu rozpoczynają się od struktury MQDLH. Należy również pamiętać, że komunikaty, które aplikacje pobierają z kolejki niedostarczonych komunikatów, mogą zostać obcięte, jeśli były pierwotnie zbyt długie dla tej kolejki.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 485. Pola w MQDLH dla MQDLH

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQDLH_ID_struktury	'DLH¬'
Wersja (numer wersji struktury)	MQDLH_VERSION_1	1
Przyczyna (komunikat o przyczynie pojawił się w kolejce niedostarczonych komunikatów)	MQRC_BRAK	0
DestQName (nazwa oryginalnej kolejki docelowej)	Brak	Pusty łańcuch lub odstępy
DestQMgrNazwa (nazwa oryginalnego docelowego menedżera kolejek)	Brak	Pusty łańcuch lub odstępy
Kodowanie (kodowanie liczbowe danych następujących po MQDLH)	Brak	0
CodedCharSetId (identyfikator zestawu znaków danych po MQDLH)	MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0
Format (nazwa formatu danych po MQDLH)	MQFMT_BRAK	Puste
PutApplTyp (typ aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów)	Brak	0
PutApplNazwa (nazwa aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów)	Brak	Pusty łańcuch lub odstępy
PutDate (data umieszczenia komunikatu w kolejce niedostarczonych komunikatów)	Brak	Pusty łańcuch lub odstępy
PutTime (czas umieszczenia komunikatu w kolejce niedostarczonych komunikatów)	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQDLH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQDLH MyDLH = {MQDLH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQDLH

```
typedef struct tagMQDLH MQDLH;
struct tagMQDLH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Reason;          /* Reason message arrived on dead-letter
                                (undelivered-message) queue */
  MQCHAR48  DestQName;       /* Name of original destination queue */
  MQCHAR48  DestQMgrName;    /* Name of original destination queue
                                manager */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQDLH */
  MQLONG    CodedCharSetId;  /* Character set identifier of data that
                                follows MQDLH */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQDLH */
  MQLONG    PutApplType;     /* Type of application that put message on
                                dead-letter (undelivered-message)
                                queue */
  MQCHAR28  PutApplName;     /* Name of application that put message on
                                dead-letter (undelivered-message)
                                queue */
  MQCHAR8   PutDate;         /* Date when message was put on dead-letter
                                (undelivered-message) queue */
  MQCHAR8   PutTime;         /* Time when message was put on the
                                dead-letter (undelivered-message)
                                queue */
};
```

Deklaracja języka COBOL dla MQDLH

```
**   MQDLH structure
  10 MQDLH.
**    Structure identifier
   15 MQDLH-STRUCID        PIC X(4).
**    Structure version number
   15 MQDLH-VERSION        PIC S9(9) BINARY.
**    Reason message arrived on dead-letter (undelivered-message) queue
   15 MQDLH-REASON         PIC S9(9) BINARY.
**    Name of original destination queue
   15 MQDLH-DESTQNAME      PIC X(48).
**    Name of original destination queue manager
   15 MQDLH-DESTQMGRNAME   PIC X(48).
**    Numeric encoding of data that follows MQDLH
   15 MQDLH-ENCODING       PIC S9(9) BINARY.
**    Character set identifier of data that follows MQDLH
   15 MQDLH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQDLH
   15 MQDLH-FORMAT         PIC X(8).
**    Type of application that put message on dead-letter
**    (undelivered-message) queue
   15 MQDLH-PUTAPPLTYPE    PIC S9(9) BINARY.
**    Name of application that put message on dead-letter
**    (undelivered-message) queue
   15 MQDLH-PUTAPPLNAME    PIC X(28).
**    Date when message was put on dead-letter (undelivered-message)
**    queue
   15 MQDLH-PUTDATE        PIC X(8).
**    Time when message was put on the dead-letter (undelivered-message)
**    queue
   15 MQDLH-PUTTIME        PIC X(8).
```

Deklaracja języka PL/I dla MQDLH

```
dcl
 1 MQDLH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Reason         fixed bin(31), /* Reason message arrived on
                                     dead-letter (undelivered-message)
                                     queue */
  3 DestQName      char(48),      /* Name of original destination
```

```
                                    queue */
  3 DestQMgrName   char(48),      /* Name of original destination queue
                                    manager */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                    follows MQDLH */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                    that follows MQDLH */
  3 Format         char(8),       /* Format name of data that follows
                                    MQDLH */
  3 PutApplType    fixed bin(31), /* Type of application that put
                                    message on dead-letter
                                    (undelivered-message) queue */
  3 PutApplName    char(28),      /* Name of application that put
                                    message on dead-letter
                                    (undelivered-message) queue */
  3 PutDate        char(8),       /* Date when message was put on
                                    dead-letter (undelivered-message)
                                    queue */
  3 PutTime        char(8);       /* Time when message was put on the
                                    dead-letter (undelivered-message)
                                    queue */
```

Deklaracja High Level Assembler dla MQDLH

```
MQDLH                 DSECT
MQDLH_STRUCID         DS   CL4   Structure identifier
MQDLH_VERSION         DS   F     Structure version number
MQDLH_REASON          DS   F     Reason message arrived on dead-letter
*                                (undelivered-message) queue
MQDLH_DESTQNAME       DS   CL48  Name of original destination queue
MQDLH_DESTQMGRNAME    DS   CL48  Name of original destination queue
*                                manager
MQDLH_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQDLH
MQDLH_CODEDCHARSETID  DS   F     Character set identifier of data that
*                                follows MQDLH
MQDLH_FORMAT          DS   CL8   Format name of data that follows MQDLH
MQDLH_PUTAPPLTYPE     DS   F     Type of application that put message on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTAPPLNAME     DS   CL28  Name of application that put message on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTDATE         DS   CL8   Date when message was put on
*                                dead-letter (undelivered-message) queue
MQDLH_PUTTIME         DS   CL8   Time when message was put on the
*                                dead-letter (undelivered-message) queue
*
MQDLH_LENGTH          EQU  *-MQDLH
                      ORG  MQDLH
MQDLH_AREA            DS   CL(MQDLH_LENGTH)
```

Deklaracja Visual Basic dla MQDLH

```
Type MQDLH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  Reason         As Long      'Reason message arrived on dead-letter'
                              '(undelivered-message) queue'
  DestQName      As String*48 'Name of original destination queue'
  DestQMgrName   As String*48 'Name of original destination queue'
                              'manager'
  Encoding       As Long      'Numeric encoding of data that follows'
                              'MQDLH'
  CodedCharSetId As Long      'Character set identifier of data that'
                              'follows MQDLH'
  Format         As String*8  'Format name of data that follows MQDLH'
  PutApplType    As Long      'Type of application that put message on'
                              'dead-letter (undelivered-message) queue'
  PutApplName    As String*28 'Name of application that put message on'
                              'dead-letter (undelivered-message) queue'
  PutDate        As String*8  'Date when message was put on dead-letter'
                              '(undelivered-message) queue'
  PutTime        As String*8  'Time when message was put on the'
                              'dead-letter (undelivered-message) queue'
End Type
```

StrucId (MQCHAR4) dla MQDLH

Jest to identyfikator struktury struktury nagłówka niedostarczonego komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQDLH_STRUC_ID.

Wartość musi być następująca:

MQDLH_ID_struktury
Identyfikator struktury nagłówka niedostarczonego komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQDLH_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQDLH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQDLH

Wersja jest numerem wersji struktury.

Wartość musi być następująca:

MQDLH_VERSION_1
Numer wersji dla struktury nagłówka niedostarczonego komunikatu.

Następująca stała określa numer wersji bieżącej:

MQDLH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka niedostarczonego komunikatu.

Wartością początkową tego pola jest MQDLH_VERSION_1.

Przyczyna (MQLONG) dla MQDLH

Pole Przyczyna identyfikuje przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów zamiast w oryginalnej kolejce docelowej.

Identyfikuje to przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów zamiast w oryginalnej kolejce docelowej. Powinna to być jedna z wartości MQFB_ * lub MQRC_ * (na przykład MQRC_Q_FULL). Szczegółowe informacje na temat wspólnych wartości MQFB_ *, które mogą wystąpić, zawiera opis pola *Feedback* w pliku “MQMD-deskryptor komunikatu” na stronie 432 .

Jeśli wartość należy do zakresu od MQFB_IMS_FIRST do MQFB_IMS_LAST, rzeczywisty kod błędu IMS można określić, odejmując wartość MQFB_IMS_ERROR od wartości pola *Reason* .

Niektóre wartości MQFB_ * występują tylko w tym polu. Odnoszą się one do komunikatów repozytorium, komunikatów wyzwalacza lub komunikatów kolejki transmisji, które zostały przesłane do kolejki niedostarczonych komunikatów. Są to:

MQFB_APPL_CANNOT_BE_STARTED (X'00000109')
Aplikacja przetwarzająca komunikat wyzwalacza nie może uruchomić aplikacji o nazwie określonej w polu *ApplId* komunikatu wyzwalacza (patrz sekcja “MQTM-komunikat wyzwalacza” na stronie 618).

W systemie z/OStransakcja CKTI CICS jest przykładem aplikacji, która przetwarza komunikaty wyzwalacza.

BŁĄD MQFB_APPL_TYPE_ERROR (X'0000010B')
Aplikacja przetwarzająca komunikat wyzwalacza nie może uruchomić aplikacji, ponieważ pole *ApplType* komunikatu wyzwalacza jest niepoprawne (patrz sekcja “MQTM-komunikat wyzwalacza” na stronie 618).

W systemie z/OStransakcja CKTI CICS jest przykładem aplikacji, która przetwarza komunikaty wyzwalacza.

MQFB_BIND_OPEN_CLUSRCVR_DEL (X'00000119')
Komunikat znajdował się w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE przeznaczona dla kolejki klastra, która została otwarta z opcją MQOO_BIND_ON_OPEN, ale zdalny kanał odbiorczy klastra, który ma być używany do przesyłania komunikatu do kolejki docelowej, został usunięty przed wysłaniem komunikatu. Ponieważ określono opcję MQOO_BIND_ON_OPEN, do przesłania komunikatu można użyć tylko kanału wybranego podczas otwierania kolejki. Ponieważ ten kanał nie jest już dostępny, komunikat jest umieszczany w kolejce niedostarczonych komunikatów.

MQFB_NOT_A_REPOSITORY_MSG (X'00000118')
Komunikat nie jest komunikatem repozytorium.

MQFB_STOPPED_BY_CHAD_EXIT (X'00000115')
Komunikat został zatrzymany przez wyjście automatycznej definicji kanału.

MQFB_STOPPED_BY_MSG_EXIT (X'0000010D')
Komunikat został zatrzymany przez wyjście komunikatu kanału.

MQFB_TM_ERROR (X'0000010A')
Pole *Format* w strukturze MQMD określa wartość MQFMT_TRIGGER, ale komunikat nie rozpoczyna się od poprawnej struktury MQTM. Na przykład mnemonik *StrucId* może być niepoprawny, *Version* może nie zostać rozpoznany lub długość komunikatu wyzwalacza może być niewystarczająca do zawarcia struktury MQTM.

W systemie z/OStransakcja CKTI CICS jest przykładem aplikacji, która przetwarza komunikaty wyzwalacza i może wygenerować ten kod sprzężenia zwrotnego.

MQFB_XMIT_Q_MSG_ERROR (X'0000010F')
Agent kanału komunikatów wykrył, że komunikat w kolejce transmisji ma niepoprawny format. Agent kanału komunikatów umieszcza komunikat w kolejce niedostarczonych komunikatów przy użyciu tego kodu sprzężenia zwrotnego.

Jedną z częstych przyczyn jest to, że komunikat został umieszczony bezpośrednio w kolejce transmisji, więc komunikat nie ma oczekiwanego nagłówka XQH. Komunikaty powinny być umieszczane w kolejce transmisji za pośrednictwem kolejki zdalnej, chyba że aplikacja buduje nagłówek MQXQH.

Wartością początkową tego pola jest MQRC_NONE.

DestQName (MQCHAR48) dla MQDLH
DestQName to nazwa kolejki komunikatów, która była oryginalnym miejscem docelowym komunikatu.

Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

DestQMgrNazwa (MQCHAR48) dla MQDLH
DestQMgrNazwa jest nazwą menedżera kolejek, który był oryginalnym miejscem docelowym komunikatu.

Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

Kodowanie (MQLONG) dla MQDLH
Kodowanie jest kodowaniem liczbowym danych, które są zgodne ze strukturą MQDLH (zwykle są to dane z oryginalnego komunikatu). Nie ma ono zastosowania do danych liczbowych w samej strukturze MQDLH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

CodedCharSetId (MQLONG) dla MQDLH
CodedCharSetId to identyfikator zestawu znaków danych przepływających przez strukturę MQDLH (zazwyczaj dane z oryginalnego komunikatu). Nie ma on zastosowania do danych znakowych w samej strukturze MQDLH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dane znakowe w danych następujących po tej strukturze znajdują się w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Pola MQCCSI_INHERIT nie można używać, jeśli wartością pola *PutApplType* w deskryptorze MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQDLH
Format jest nazwą formatu danych, które są zgodne ze strukturą MQDLH (zwykle są to dane z oryginalnego komunikatu).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w MQMD.

Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

PutApplTyp (MQLONG) dla MQDLH
PutApplTyp aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów.

To pole ma takie samo znaczenie jak pole *PutApplType* w deskryptorze MQMD komunikatu (szczegółowe informacje zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432).

Jeśli menedżer kolejek przekierowuje komunikat do kolejki niedostarczonych komunikatów, parametr *PutApplType* ma wartość MQAT_QMGR.

Wartością początkową tego pola jest 0.

PutApplNazwa (MQCHAR28) dla MQDLH
PutApplNazwa jest nazwą aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów.

Format nazwy zależy od pola *PutApplType* . Format może różnić się w zależności od wersji. Patrz opis pola *PutApplName* w sekcji “MQMD-deskryptor komunikatu” na stronie 432.

Jeśli menedżer kolejek przekierowuje komunikat do kolejki niedostarczonych komunikatów, produkt *PutApplName* zawiera pierwsze 28 znaków nazwy menedżera kolejek, w razie potrzeby dopełnione odstępami.

Długość tego pola jest określona przez wartość MQ_PUT_APPL_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 28 pustych znaków w innych językach programowania.

PutDate (MQCHAR8) dla MQDLH
PutDate to data umieszczenia komunikatu w kolejce niedostarczonych komunikatów.

Format używany dla daty wygenerowania tego pola przez menedżer kolejek to:

- RRRRMMDD

gdzie znaki reprezentują:

rrrr
rok (cztery cyfry)

MM
miesiąc roku (od 01 do 12)

DD
dzień miesiąca (od 01 do 31)

Czas Greenwich (Greenwich Mean Time-GMT) jest używany w polach *PutDate* i *PutTime* , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na czas GMT.

Długość tego pola jest określona przez wartość MQ_PUT_DATE_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i osiem znaków odstępu w innych językach programowania.

PutTime (MQCHAR8) dla MQDLH

PutTime to czas umieszczenia komunikatu w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Format używany dla czasu wygenerowania tego pola przez menedżer kolejek to:

- GGMMSSTH

gdzie znaki reprezentują:

GG
godzin (od 00 do 23)

MM
minuty (od 00 do 59)

SS
sekundy (od 00 do 59; patrz uwaga)

T
dziesiąte części sekundy (od 0 do 9)

H
setne sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z bardzo dokładnym standardem czasu, w rzadkich przypadkach możliwe jest zwrócenie wartości 60 lub 61 dla sekund w pliku *PutTime*. Dzieje się tak, gdy sekundy przestępne są wstawiane do globalnego standardu czasu.

Czas Greenwich (Greenwich Mean Time-GMT) jest używany w polach *PutDate* i *PutTime* , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na czas GMT.

Długość tego pola jest określona przez wartość MQ_PUT_TIME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i osiem znaków odstępu w innych językach programowania.

MQDMHO-opcje usuwania uchwytu komunikatu

Struktura **MQDMHO** umożliwia aplikacjom określanie opcji sterujących usuwaniem uchwytów komunikatów. Struktura jest parametrem wejściowym wywołania **MQDLTMH** .

Zestaw znaków i kodowanie

Dane w pliku **MQDMHO** muszą znajdować się w zestawie znaków aplikacji i kodowania aplikacji (**MQENC_NATIVE**).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 486. Pola w MQDMHO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQDMHO_STRUC_ID	'DMHO'
Wersja (numer wersji struktury)	MQDMHO_VERSION_1	1
Opcje (opcje)	MQDMHO_NONE	0

Uwagi:

1. W języku programowania C: zmienna makraParametr MQDMHO_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQDMHO MyDMHO = {MQDMHO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQDMHO

```
typedef struct tagMQDMHO;
struct tagMQDMHO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQDLTMH */
};
```

Deklaracja języka COBOL dla MQDMHO

```
**   MQDMHO structure
  10 MQDMHO.
**    Structure identifier
   15 MQDMHO-STRUCID       PIC X(4).
**    Structure version number
   15 MQDMHO-VERSION       PIC S9(9) BINARY.
**    Options that control the action of MQDLTMH
   15 MQDMHO-OPTIONS       PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQDMHO

```
dcl
 1 MQDMHO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQDLTMH */
```

Deklaracja High Level Assembler dla MQDMHO

```
MQDMHO                DSECT
MQDMHO_STRUCID        DS   CL4   Structure identifier
MQDMHO_VERSION        DS   F     Structure version number
MQDMHO_OPTIONS        DS   F     Options that control the action of
*                                MQDLTMH
MQDMHO_LENGTH         EQU  *-MQDMHO
MQDMHO_AREA           DS   CL(MQDMHO_LENGTH)
```

StrucId (MQCHAR4) dla MQDMHO

Jest to identyfikator struktury struktury opcji usuwania uchwytu komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQDMHO_STRUC_ID.

Wartość musi być następująca:

MQDMHO_STRUC_ID

Identyfikator struktury opcji usuwania uchwytu komunikatu.

Dla języka programowania C zdefiniowana jest również stała **MQDMHO_STRUC_ID_ARRAY** . Ma taką samą wartość jak **MQDMHO_STRUC_ID**, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQDMHO

Jest to numer wersji struktury; wartość musi być następująca:

MQDMHO_VERSION_1
Version-1 usunięcie struktury opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej:

MQDMHO_CURRENT_VERSION
Bieżąca wersja struktury opcji usuwania uchwytu komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **MQDMHO_VERSION_1**.

Opcje (MQLONG) dla MQDMHO

Wartość musi być następująca:

MQDMHO_NONE

Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **MQDMHO_NONE**.

MQDMPO-opcje właściwości usuwania komunikatu

Struktura MQDMPO umożliwia aplikacjom określanie opcji sterujących sposobem usuwania właściwości komunikatów. Struktura jest parametrem wejściowym wywołania MQDLTMP.

Zestaw znaków i kodowanie

Dane w obiekcie MQDMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 487. Pola w MQDPMO

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	Identyfikator struktury MQDMPO_STRUC_ID	'DMPO'
Wersja (numer wersji struktury)	MQDMPO_VERSION_1	1
Opcje (opcje sterujące działaniem komendy MQDMPO)	Opcje sterujące działaniem komendy MQDLTMP	MQDMPO_BRAK

Tabela 487. Pola w MQDPMO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**

Uwagi:

1. W języku programowania C: zmienna makraParametr MQDMPO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQDMPO MyDMPO = {MQDMPO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQDMPO

```
typedef struct tagMQDMPO MQDMPO;
struct tagMQDMPO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQDLTMP */
};
```

Deklaracja języka COBOL dla MQDMPO

```
**   MQDMPO structure
  10 MQDMPO.
**    Structure identifier
   15 MQDMPO-STRUCID               PIC X(4).
**    Structure version number
   15 MQDMPO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQDLTMP
   15 MQDMPO-OPTIONS               PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQDMPO

```
Dcl
 1 MQDMPO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action
                                      of MQDLTMP */
```

Deklaracja High Level Assembler dla MQDMPO

```
MQDMPO                   DSECT
MQDMPO_STRUCID           DS   CL4  Structure identifier
MQDMPO_VERSION           DS   F    Structure version number
MQDMPO_OPTIONS           DS   F    Options that control the
*                                   action of MQDLTMP
MQDMPO_LENGTH            EQU  *-MQDMPO
MQDMPO_AREA              DS   CL(MQDMPO_LENGTH)
```

StrucId (MQCHAR4) dla MQDMPO

Jest to identyfikator struktury struktury opcji usuwania właściwości komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQDMPO_STRUC_ID.

Wartość musi być następująca:

Identyfikator struktury MQDMPO_STRUC_ID
Identyfikator struktury opcji usuwania właściwości komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQDMPO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQDMPO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQDMPO

Usuń strukturę opcji właściwości komunikatu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQDMPO_VERSION_1
Numer wersji struktury opcji usuwania właściwości komunikatu.

Następująca stała określa numer wersji bieżącej:

MQDMPO_CURRENT_VERSION
Bieżąca wersja struktury opcji usuwania właściwości komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQDMPO_VERSION_1.

Opcje (MQLONG) dla MQDMPO

Usuń strukturę opcji właściwości komunikatu-pole Opcje

Opcje położenia: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości.

MQDMPO_DEL_FIRST
Usuwa pierwszą właściwość, która jest zgodna z podaną nazwą.

MQDMPO_DEL_PROP_UNDER_CURSOR,
Powoduje usunięcie właściwości wskazywanej przez kursor właściwości, czyli właściwości, do której ostatnio wysłano zapytanie przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany. Jest ona również resetowana, jeśli uchwyt komunikatu jest określony w polu *MsgHandle* w strukturze MQGMO w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze ustanowiony, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i przyczyną MQRC_PROPERTY_NOT_AVAILABLE. Jeśli właściwość wskazywanej przez kursor właściwości została już usunięta, wywołanie również kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i przyczyną MQRC_PROPERTY_NOT_AVAILABLE.

Jeśli żadna z tych opcji nie jest wymagana, można użyć następującej opcji:

MQDMPO_BRAK
Nie określono żadnych opcji.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest MQDMPO_DEL_FIRST.

MQEPH-wbudowany nagłówek PCF

Struktura MQEPH opisuje dodatkowe dane, które są obecne w komunikacie, gdy komunikat jest komunikatem w formacie programowalnej komendy (PCF). Pole *PCFHeader* definiuje parametry PCF, które są zgodne z tą strukturą i umożliwia śledzenie danych komunikatu PCF z innymi nagłówkami.

Nazwa formatu

MQFMT_EMBEDDED_PCF

Zestaw znaków i kodowanie

Dane w programie MQEPH muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE.

Ustaw zestaw znaków i kodowanie MQEPH w polach *CodedCharSetId* i *Encoding* w strukturze MQMD (jeśli struktura MQEPH znajduje się na początku danych komunikatu) lub w strukturze nagłówka poprzedzającej strukturę MQEPH (wszystkie inne przypadki).

Użycie

Struktur MQEPH nie można używać do wysyłania komend do serwera komend lub innego serwera akceptującego PCF menedżera kolejek.

Podobnie serwer komend lub dowolny inny serwer akceptujący PCF menedżera kolejek nie generuje odpowiedzi ani zdarzeń zawierających struktury MQEPH.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 488. Pola w MQEPH dla MQEPH

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	ID_STRUKTURY_MQEPH _STRUCT	'EPH¬'
Wersja (numer wersji struktury)	MQEPH_VERSION_1	1
StrucLength (długość struktury MQEPH oraz struktury MQCFH i kolejnych struktur parametrów)	MQEPH_STRUC_LENGT H_FIXED	68
Kodowanie (kodowanie liczbowe danych po ostatniej strukturze parametru PCF)	Brak	0
CodedCharSetId (identyfikator zestawu znaków danych, który występuje po ostatniej strukturze parametru PCF)	MQCCSI_XX_ENCODE_C ASE_ONE niezdefiniowany	0
Format (nazwa formatu danych po ostatniej strukturze parametru PCF)	MQFMT_BRAK	Puste
Flagi (flags)	MQEPH_BRAK	0
PCFHeader (nagłówek programowalnego formatu komend (PCF))	Nazwy i wartości zdefiniowane w pliku Tabela 489 na stronie 376	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQEPH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQEPH MyEPH = {MQEPH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQEPH

```
typedef struct tagMQEPH MQEPH;
struct tagMQDH {
```

```
  MQCHAR4  StrucId;          /* Structure identifier */
  MQLONG   Version;          /* Structure version number */
  MQLONG   StrucLength;      /* Total length of MQEPH including the MQCFH
                                and parameter structures that follow it */
  MQLONG   Encoding;         /* Numeric encoding of data that follows last
                                PCF parameter structure */
  MQLONG   CodedCharSetId;   /* Character set identifier of data that
                                follows last PCF parameter structure */
  MQCHAR8  Format;           /* Format name of data that follows last PCF
                                parameter structure */
  MQLONG   Flags;            /* Flags */
  MQCFH    PCFHeader;        /* Programmable command format header */
 };
```

Deklaracja języka COBOL dla MQEPH

```
**   MQEPH structure
  10 MQEPH.
**    Structure identifier
   15 MQEPH-STRUCID         PIC X(4).
**    Structure version number
   15 MQEPH-VERSION         PIC S9(9) BINARY.
**    Total length of MQEPH structure including the MQCFH
**    and parameter structures that follow it
   15 MQEPH-STRUCLENGTH     PIC S9(9) BINARY.
**    Numeric encoding of data that follows last
**    PCF structure
   15 MQEPH-ENCODING        PIC S9(9) BINARY.
**    Character set identifier of data that
**    follows last PCF parameter structure
   15 MQEPH-CODEDCHARSETID  PIC S9(9) BINARY.
**    Format name of data that follows last PCF
**    parameter structure
   15 MQEPH-FORMAT          PIC X(8).
**    Flags
   15 MQEPH-FLAGS           PIC S9(9) BINARY.
**    Programmable command format header
   15 MQEPH-PCFHEADER.
**      Structure type
     20 MQEPH-PCFHEADER-TYPE           PIC S9(9) BINARY.
**      Structure length
     20 MQEPH-PCFHEADER-STRUCLENGTH    PIC S9(9) BINARY.
**      Structure version number
     20 MQEPH-PCFHEADER-VERSION        PIC S9(9) BINARY.
**      Command identifier
     20 MQEPH-PCFHEADER-COMMAND        PIC S9(9) BINARY.
**      Message sequence number
     20 MQEPH-PCFHEADER-MSGSEQNUMBER   PIC S9(9) BINARY.
**      Control options
     20 MQEPH-PCFHEADER-CONTROL        PIC S9(9) BINARY.
**      Completion code
     20 MQEPH-PCFHEADER-COMPCODE       PIC S9(9) BINARY.
**      Reason code qualifying completion code
     20 MQEPH-PCFHEADER-REASON         PIC S9(9) BINARY.
**      Count of parameter structures
     20 MQEPH-PCFHEADER-PARAMETERCOUNT PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQEPH

```
dcl
 1 MQEPH based,
  3 StrucId         char(4),       /* Structure identifier */
  3 Version         fixed bin(31), /* Structure version number */
  3 StrucLength     fixed bin(31), /* Total Length of MQEPH including the
                                      MQCFH and parameter structures that
                                      follow it
  3 Encoding        fixed bin(31), /* Numeric encoding of data that follows
                                      last PCF parameter structure
  3 CodedCharSetId  fixed bin(31), /* Character set identifier of data that
                                      follows last PCF parameter structure
  3 Format          char(8),       /* Format name of data that follows last
                                      PCF parameter structure */
  3 Flags           fixed bin(31), /* Flags */
  3 PCFHeader,                     /* Programmable command format header
   5 Type           fixed bin(31), /* Structure type */
   5 StrucLength    fixed bin(31), /* Structure length */
   5 Version        fixed bin(31), /* Structure version number */
```

```
   5 Command        fixed bin(31), /* Command identifier */
   5 MsgseqNumber   fixed bin(31), /* Message sequence number */
   5 Control        fixed bin(31), /* Control options */
   5 CompCode       fixed bin(31), /* Completion code */
   5 Reason         fixed bin(31), /* Reason code qualifying completion code */
   5 ParameterCount fixed bin(31); /* Count of parameter structures */
```

Deklaracja High Level Assembler dla MQEPH

```
MQEPH                           DSECT
MQEPH_STRUCID                   DS   CL4   Structure identifier
MQEPH_VERSION                   DS   F     Structure version number
MQEPH_STRUCLENGTH               DS   F     Total length of MQEPH including the
*                                          MQCFH and parameter structures that
                                           follow it
MQEPH_ENCODING                  DS   F     Numeric encoding of data that follows
*                                          last PCF parameter structure
MQEPH_CODEDCHARSETID            DS   F     Character set identifier of data that
*                                          follows last PCF parameter structure
MQEPH_FORMAT                    DS   CL8   Format name of data that follows last
*                                          PCF parameter structure
MQEPH_FLAGS                     DS   F     Flags
MQEPH_PCFHEADER                 DS   0F    Force fullword alignment
MQEPH_PCFHEADER_TYPE            DS   F     Structure type
MQEPH_PCFHEADER_STRUCLENGTH     DS   F     Structure length
MQEPH_PCFHEADER_VERSION         DS   F     Structure version number
MQEPH_PCFHEADER_COMMAND         DS   F     Command identifier
MQEPH_PCFHEADER_MSGSEQNUMBER    DS   F     Structure length
MQEPH_PCFHEADER_CONTROL         DS   F     Control options
MQEPH_PCFHEADER_COMPCODE        DS   F     Completion code
MQEPH_PCFHEADER_REASON          DS   F     Reason code qualifying completion code
MQEPH_PCFHEADER_PARAMETER COUNT DS   F     Count of parameter structures
MQEPH_PCFHEADER_LENGTH          EQU  *-MQEPH_PCFHEADER
                                ORG  MQEPH_PCFHEADER
MQEPH_PCFHEADER_AREA            DS   CL(MQEPH_PCFHEADER_LENGTH)
*
MQEPH_LENGTH                    EQU  *-MQEPH
                                ORG  MQEPH
MQEPH_AREA                      DS   CL(MQEPH_LENGTH)
```

Deklaracja Visual Basic dla MQEPH

```
Type MQEPH
  StrucId         As String*4 'Structure identifier'
  Version         As Long     'Structure version number'
  StrucLength     As Long     'Total length of MQEPH structure including the MQCFH'
                              'and parameter structures that follow it'
  Encoding        As Long     'Numeric encoding of data that follows last'
                              'PCF parameter structure'
  CodedCharSetId  As Long     'Character set identifier of data that'
                              'follows last PCF parameter structure'
  Format          As String*8 'Format name of data that follows last PCF'
                              'parameter structure'
  Flags           As Long     'Flags'
  PCFHeader       As MQCFH    'Programmable command format header'
 End Type

Global MQEPH_DEFAULT As MQEPH
```

StrucId (MQCHAR4) dla MQEPH

Jest to identyfikator struktury osadzonej struktury nagłówka PCF. Jest to zawsze pole wejściowe. Jego wartość to MQEPH_STRUC_ID.

Wartość musi być następująca:

ID_STRUKTURY_MQEPH_STRUCT
Identyfikator osadzonej struktury nagłówka PCF.

Dla języka programowania C zdefiniowana jest również stała MQEPH_STRUC_ID_ARRAY. Ma taką samą wartość jak MQEPH_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQEPH

Wartość musi być następująca:

MQEPH_VERSION_1
Numer wersji dla osadzonej struktury nagłówka PCF.

Następująca stała określa numer wersji bieżącej:

MQCFH_VERSION_3
Bieżąca wersja wbudowanej struktury nagłówka PCF.

Wartością początkową tego pola jest MQEPH_VERSION_1.

StrucLength (MQLONG) dla MQEPH

Jest to ilość danych poprzedzających następną strukturę nagłówka. i zapewnia Klientowi:

- Długość nagłówka MQEPH
- Długość wszystkich parametrów PCF następujących po nagłówku
- Puste dopełnianie po tych parametrach

StrucLength musi być wielokrotnością 4.

Część struktury o stałej długości jest zdefiniowana przez parametr MQEPH_STRUC_LENGTH_FIXED.

Wartością początkową tego pola jest 68.

Kodowanie (MQLONG) dla MQEPH

Jest to kodowanie liczbowe danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF. Nie ma ono zastosowania do danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest 0.

CodedCharSetId (MQLONG) dla MQEPH

Jest to identyfikator zestawu znaków danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF. Nie ma on zastosowania do danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQEPH

Jest to nazwa formatu danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF.

Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQEPH

Dozwolone są następujące wartości:

MQEPH_BRAK
Nie określono żadnych opcji. Parametr MQEPH_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta stała była używana z żadną inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

MQEPH_CCSID_EMBEDDED
Zestaw znaków parametrów zawierających dane znakowe jest określany indywidualnie w polu CodedCharSetId w każdej strukturze. Zestaw znaków pól StrucId i Format jest definiowany przez pole CodedCharSetId w strukturze nagłówka poprzedzającej strukturę MQEPH lub przez pole CodedCharSetId w strukturze MQMD, jeśli MQEPH znajduje się na początku komunikatu.

Wartością początkową tego pola jest MQEPH_NONE.

PCFHeader (MQCFH) dla MQEPH

Jest to nagłówek formatu komend programowalnych (PCF), definiujący parametry PCF zgodne ze strukturą MQEPH. Umożliwia to śledzenie danych komunikatu PCF z innymi nagłówkami.

Nagłówek PCF jest początkowo zdefiniowany z następującymi wartościami:

Tabela 489. Pola w MQCFH		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
Type	MQCFT_BRAK	0
StrucLength	MQCFH_STRUC_LENGTH	36
Version	MQCFH_VERSION_3	3
StrucLength	Brak	0
Command	MQCMD_NONE	0
MsgSeqNumber	Brak	1
Control	MQCFC_LAST	1
CompCode	MQCC_OK	0
Reason	MQRC_BRAK	0
ParameterCount	Brak	0

Aplikacja musi zmienić typ Type z MQCFT_NONE na poprawny typ struktury na potrzeby używania osadzonego nagłówka PCF.

MQGMO-opcje pobierania komunikatów

Struktura MQGMO umożliwia aplikacji sterowanie sposobem usuwania komunikatów z kolejek. Struktura jest parametrem wejścia/wyjścia w wywołaniu MQGET.

Wersja

Bieżąca wersja MQGMO to MQGMO_VERSION_4. Niektóre pola są dostępne tylko w niektórych wersjach MQGMO. Jeśli konieczne jest przeniesienie aplikacji między kilkoma środowiskami, należy upewnić się, że wersja MQGMO jest spójna we wszystkich środowiskach. Pola, które istnieją tylko w określonych wersjach struktury, są identyfikowane jako takie w pliku "MQGMO-opcje pobierania komunikatów" na stronie 376 i w opisach pól.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję produktu MQGMO obsługiwaną przez środowisko, ale z wartością początkową pola *Version* ustawioną na wartość MQGMO_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , należy ustawić w polu *Version* numer wersji wymaganej wersji.

Zestaw znaków i kodowanie

Dane w obiekcie MQGMO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 490. Pola w MQGMO dla MQGMO		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQGMO_ID_struktury	'GMO¬'
Wersja (numer wersji struktury)	MQGMO_VERSION_1	1
MQGMO-pole Opcje (opcje sterujące działaniem komendy MQGET)	MQGMO_NO_WAIT	0
WaitInterval (odstęp czasu oczekiwania)	Brak	0
Signal1 (sygnał)	Brak	Pusty wskaźnik na z/OS ; 0 w przeciwnym razie
Signal2 (identyfikator sygnału)	Brak	0
ResolvedQName (rozstrzygnięta nazwa kolejki docelowej)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.		
MatchOptions (opcje sterujące kryteriami wyboru używanymi dla wywołania MQGET)	MQMO_MATCH_MSG_ID + MQMO_MATCH_CORREL_ID	3
GroupStatus (flaga wskazująca, czy pobrany komunikat należy do grupy)	GRUPA MQGS_NOT_IN_	'¬'
SegmentStatus (flaga wskazująca, czy pobrany komunikat jest segmentem komunikatu logicznego)	MQSS_NOT_A_SEGMENT (MQSS_NOT_A)	'¬'
Segmentacja (flaga wskazująca, czy dla pobieranego komunikatu dozwolona jest dalsza segmentacja)	MQSEG_INHIBITED	'¬'
Reserved1 (zarezerwowany)	Brak	'¬'
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_3.		
MsgToken (znacznik komunikatu)	MQMTOK_BRAK	Wartości null
ReturnedLength (długość w bajtach zwróconych danych komunikatu)	MQRL_UNDEFINED (niezdefiniowana)	-1
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQGMO_VERSION_4.		
Reserved2 (zarezerwowany)	Brak	'¬'
MsgHandle (uchwyt do komunikatu, który ma zostać zapełniony właściwościami komunikatu pobieranego z kolejki)	MQHM_NONE	0

Tabela 490. Pola w MQGMO dla MQGMO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQGMO_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQGMO MyGMO = {MQGMO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQGMO

```
typedef struct tagMQGMO MQGMO;
struct tagMQGMO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of */
                                MQGET */
  MQLONG    WaitInterval;    /* Wait interval */
  MQLONG    Signal1;         /* Signal */
  MQLONG    Signal2;         /* Signal identifier */
  MQCHAR48  ResolvedQName;   /* Resolved name of destination queue */
  /* Ver:1 */
  MQLONG    MatchOptions;    /* Options controlling selection */
                                criteria used for MQGET */
  MQCHAR    GroupStatus;     /* Flag indicating whether message */
                                retrieved is in a group */
  MQCHAR    SegmentStatus;   /* Flag indicating whether message */
                                retrieved is a segment of a logical */
                                message */
  MQCHAR    Segmentation;    /* Flag indicating whether further */
                                segmentation is allowed for the */
                                message retrieved */
  MQCHAR    Reserved1;       /* Reserved */
  /* Ver:2 */
  MQBYTE16  MsgToken;        /* Message token */
  MQLONG    ReturnedLength;  /* Length of message data returned */
                                (bytes) */
  /* Ver:3 */
  MQLONG    Reserved2;       /* Reserved */
  MQHMSG    MsgHandle;       /* Message handle */
  /* Ver:4 */
};
```

Uwaga: W systemie z/OSpole *Signal1* jest zadeklarowane jako PMQLONG.

Deklaracja języka COBOL dla MQGMO

```
**   MQGMO structure
  10 MQGMO.
**    Structure identifier
   15 MQGMO-STRUCID        PIC X(4).
**    Structure version number
   15 MQGMO-VERSION        PIC S9(9) BINARY.
**    Options that control the action of MQGET
   15 MQGMO-OPTIONS        PIC S9(9) BINARY.
**    Wait interval
   15 MQGMO-WAITINTERVAL   PIC S9(9) BINARY.
**    Signal
   15 MQGMO-SIGNAL1        PIC S9(9) BINARY.
**    Signal identifier
   15 MQGMO-SIGNAL2        PIC S9(9) BINARY.
```

```
**    Resolved name of destination queue
   15 MQGMO-RESOLVEDQNAME  PIC X(48).
**    Options controlling selection criteria used for MQGET
   15 MQGMO-MATCHOPTIONS   PIC S9(9) BINARY.
**    Flag indicating whether message retrieved is in a group
   15 MQGMO-GROUPSTATUS    PIC X.
**    Flag indicating whether message retrieved is a segment of a
**    logical message
   15 MQGMO-SEGMENTSTATUS  PIC X.
**    Flag indicating whether further segmentation is allowed for the
**    message retrieved
   15 MQGMO-SEGMENTATION   PIC X.
**    Reserved
   15 MQGMO-RESERVED1      PIC X.
**    Message token
   15 MQGMO-MSGTOKEN       PIC X(16).
**    Length of message data returned (bytes)
   15 MQGMO-RETURNEDLENGTH PIC S9(9) BINARY.
**    Reserved
   15 MQGMO-RESERVED2      PIC S9(9) BINARY.
**    Message handle
   15 MQGMO-MSGHANDLE      PIC S9(18) BINARY.
```

Uwaga: W systemie z/OSpole *Signal1* jest zadeklarowane jako POINTER.

Deklaracja języka PL/I dla MQGMO

```
dcl
 1 MQGMO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of
                                     MQGET */
  3 WaitInterval   fixed bin(31), /* Wait interval */
  3 Signal1        fixed bin(31), /* Signal */
  3 Signal2        fixed bin(31), /* Signal identifier */
  3 ResolvedQName  char(48),      /* Resolved name of destination
                                     queue */
  3 MatchOptions   fixed bin(31), /* Options controlling selection
                                     criteria used for MQGET */
  3 GroupStatus    char(1),       /* Flag indicating whether message
                                     retrieved is in a group */
  3 SegmentStatus  char(1),       /* Flag indicating whether message
                                     retrieved is a segment of a logical
                                     message */
  3 Segmentation   char(1),       /* Flag indicating whether further
                                     segmentation is allowed for the
                                     message retrieved */
  3 Reserved1      char(1),       /* Reserved */
  3 MsgToken       char(16),      /* Message token */
  3 ReturnedLength fixed bin(31); /* Length of message data returned
                                     (bytes) */
  3 Reserved2      fixed bin(31); /* Reserved */
  3 MsgHandle      fixed bin(63); /* Message handle */
```

Uwaga: W systemie z/OSpole *Signal1* jest zadeklarowane jako `pointer`.

Deklaracja High Level Assembler dla MQGMO

```
MQGMO                 DSECT
MQGMO_STRUCID         DS   CL4   Structure identifier
MQGMO_VERSION         DS   F     Structure version number
MQGMO_OPTIONS         DS   F     Options that control the action of
*                                MQGET
MQGMO_WAITINTERVAL    DS   F     Wait interval
MQGMO_SIGNAL1         DS   F     Signal
MQGMO_SIGNAL2         DS   F     Signal identifier
MQGMO_RESOLVEDQNAME   DS   CL48  Resolved name of destination queue
MQGMO_MATCHOPTIONS    DS   F     Options controlling selection criteria
*                                used for MQGET
MQGMO_GROUPSTATUS     DS   CL1   Flag indicating whether message
*                                retrieved is in a group
MQGMO_SEGMENTSTATUS   DS   CL1   Flag indicating whether message
*                                retrieved is a segment of a logical
*                                message
MQGMO_SEGMENTATION    DS   CL1   Flag indicating whether further
*                                segmentation is allowed for the message
```

```
*                               retrieved
MQGMO_RESERVED1       DS   CL1   Reserved
MQGMO_MSGTOKEN        DS   XL16  Message token
MQGMO_RETURNEDLENGTH  DS   F     Length of message data returned (bytes)
MQGMO_RESERVED2       DS   F     Reserved
MQGMO_MSGHANDLE       DS   D     Message handle
MQGMO_LENGTH          EQU  *-MQGMO
                      ORG  MQGMO
MQGMO_AREA            DS   CL(MQGMO_LENGTH)
```

Deklaracja High Level Assembler dla MQGMO

```
Type MQGMO
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  Options        As Long      'Options that control the action of MQGET'
  WaitInterval   As Long      'Wait interval'
  Signal1        As Long      'Signal'
  Signal2        As Long      'Signal identifier'
  ResolvedQName  As String*48 'Resolved name of destination queue'
  MatchOptions   As Long      'Options controlling selection criteria'
                              'used for MQGET'
  GroupStatus    As String*1  'Flag indicating whether message'
                              'retrieved is in a group'
  SegmentStatus  As String*1  'Flag indicating whether message'
                              'retrieved is a segment of a logical'
                              'message'
  Segmentation   As String*1  'Flag indicating whether further'
                              'segmentation is allowed for the message'
                              'retrieved'
  Reserved1      As String*1  'Reserved'
  MsgToken       As MQBYTE16  'Message token'
  ReturnedLength As Long      'Length of message data returned (bytes)'
End Type
```

PROPCTL opcje kanału dla MQGMO

Atrybut kanału **PROPCTL** służy do określania, które właściwości komunikatu są dołączane do komunikatu wysyłanego z menedżera kolejek systemu IBM MQ 9.3 do partnerskiego menedżera kolejek z wcześniejszej wersji produktu IBM MQ.

Tabela 491. Ustawienia atrybutu właściwości komunikatu kanału

PROPCTL	Opis
ALL	Tej opcji należy użyć, jeśli aplikacje połączone z partnerskim menedżerem kolejek z wcześniejszej wersji mogą przetwarzać dowolne właściwości umieszczone w komunikacie przez aplikację IBM MQ 9.3 . Wszystkie właściwości są wysyłane do partnerskiego menedżera kolejek oprócz wszystkich par nazwa-wartość umieszczonych w MQRFH2. Należy wziąć pod uwagę dwie kwestie związane z projektowaniem aplikacji: 1. Aplikacja połączona z partnerskim menedżerem kolejek musi mieć możliwość przetwarzania komunikatów zawierających nagłówki MQRFH2 wygenerowane w menedżerze kolejek systemu IBM MQ 9.3 . 2. Aplikacja połączona z partnerskim menedżerem kolejek musi poprawnie przetworzyć nowe właściwości komunikatu oznaczone flagą MQPD_SUPPORT_REQUIRED . Jeśli ustawiona jest opcja kanału ALL , aplikacje JMS mogą współdziałać między produktem IBM MQ 9.3 a wcześniejszą wersją, korzystając z tego kanału. Nowe aplikacje IBM MQ 9.3 używające właściwości komunikatu mogą współdziałać z aplikacjami we wcześniejszej wersji, w zależności od sposobu, w jaki aplikacja we wcześniejszej wersji obsługuje nagłówki MQRFH2 .

Tabela 491. Ustawienia atrybutu właściwości komunikatu kanału (kontynuacja)

PROPCTL	**Opis**
COMPAT	Ta opcja służy do wysyłania właściwości komunikatu do aplikacji połączonych z partnerskim menedżerem kolejek wcześniejszej wersji w niektórych przypadkach, ale nie do wszystkich. Właściwości komunikatu są wysyłane tylko wtedy, gdy spełnione są dwa warunki: 1. Żadna właściwość nie może być oznaczona jako wymagająca przetwarzania właściwości komunikatu. 2. Co najmniej jedna z właściwości komunikatu musi znajdować się w "zarezerwowanym" folderze; patrz Uwaga. Dzięki ustawieniu opcji kanału COMPAT aplikacje JMS mogą współdziałać między programem IBM MQ 9.3 a wcześniejszą wersją, korzystając z tego kanału. Kanał nie jest dostępny dla każdej aplikacji korzystającej z właściwości komunikatu, tylko dla tych aplikacji, które używają zarezerwowanych folderów. Reguły dotyczące tego, czy komunikat lub właściwość jest wysyłana, są następujące: 1. Jeśli komunikat ma właściwości, ale żadna z nich nie jest powiązana z "zarezerwowanym" folderem, nie są wysyłane żadne właściwości komunikatu. 2. Jeśli dowolna właściwość komunikatu została utworzona w "zarezerwowanym" folderze właściwości, wysyłane są wszystkie właściwości komunikatu powiązane z tym komunikatem. Jednakże: a. Jeśli dowolna z właściwości komunikatu jest oznaczona jako wymagana, MQPD_SUPPORT_REQUIRED lub MQPD_SUPPORT_REQUIRED_IF_LOCAL, cały komunikat jest odrzucany. Jest on zwracany, usuwany lub wysyłany do kolejki niedostarczonych komunikatów zgodnie z wartością opcji raportu. b. Jeśli żadne właściwości komunikatu nie są oznaczone jako wymagane, pojedyncza właściwość może nie zostać wysłana. Jeśli którekolwiek z pól deskryptora właściwości komunikatu są ustawione na wartości inne niż domyślne, pojedyncza właściwość nie jest wysyłana. Komunikat jest nadal wysyłany. Przykładem wartości pola deskryptora właściwości innej niż domyślna jest MQPD_USER_CONTEXT. **Uwaga:** Nazwy "zastrzeżonych" folderów rozpoczynają się od mcd., jms., usr.lub mqext.. Te foldery są tworzone dla aplikacji korzystających z interfejsu JMS . W IBM MQ 9.3 wszystkie pary nazwa-wartość, które są umieszczane w tych folderach, są traktowane jako właściwości komunikatu. Właściwości komunikatu są wysyłane w nagłówku MQRFH2 , oprócz wszystkich par nazwa-wartość umieszczonych w nagłówku MQRFH2 . Wszystkie pary nazwa-wartość umieszczone w nagłówku MQRFH2 są wysyłane, dopóki komunikat nie zostanie odrzucony.
Brak	Użyj tej opcji, aby uniemożliwić wysyłanie właściwości komunikatów do aplikacji połączonych z partnerskim menedżerem kolejek wcześniejszej wersji. MQRFH2 , który zawiera pary nazwa-wartość i właściwości komunikatu, jest nadal wysyłany, ale tylko z parami nazwa-wartość. Po ustawieniu opcji kanału NONE komunikat JMS jest wysyłany jako JMSTextMessage lub JMSBytesMessage bez żadnych właściwości komunikatu JMS . Jeśli istnieje możliwość, że aplikacja we wcześniejszej wersji zignoruje wszystkie właściwości ustawione w aplikacji IBM MQ 9.3 , może współdziałać z tą aplikacją.

PROPCTL opcje kolejki MQGMO

Atrybut kolejki **PROPCTL** służy do sterowania sposobem zwracania właściwości komunikatu do aplikacji, która wywołuje funkcję **MQGET** bez ustawiania opcji właściwości komunikatu **MQGMO** .

Tabela 492. Ustawienia atrybutu właściwości komunikatu kolejki

PROPCTL	Opis
ALL	Opcja ALL umożliwia różnym aplikacjom odczytywanie komunikatów z tej samej kolejki na różne sposoby. • Aplikacja, która została zmigrowana bez zmian z wcześniejszej wersji, może kontynuować bezpośrednie odczytywanie pliku MQRFH2 . Właściwości są bezpośrednio dostępne w nagłówku MQRFH2 . Należy zmodyfikować aplikację tak, aby obsługiwała nowe właściwości i nowe atrybuty właściwości. Zmiany w układzie i liczbie nagłówków MQRFH2 mogą mieć wpływ na aplikację. Niektóre atrybuty folderu mogą zostać usunięte lub program IBM MQ zgłosi błąd w układzie nagłówka MQRFH2 , który zignorował we wcześniejszej wersji. • Nowa lub zmieniona aplikacja może użyć właściwości komunikatu MQI w celu zapytania o właściwości komunikatu i odczytywania par nazwa-wartość bezpośrednio w nagłówku MQRFH2 . Wszystkie właściwości w komunikacie są zwracane do aplikacji. • Jeśli aplikacja wywołuje funkcję MQCRTMH w celu utworzenia uchwytu komunikatu, musi wysłać zapytanie do właściwości komunikatu przy użyciu funkcji MQINQMP. Pary nazwa-wartość, które nie są właściwościami komunikatu, pozostają w pliku MQRFH2, który jest pozbawiony właściwości komunikatu. • Jeśli aplikacja nie utworzy uchwytu komunikatu, wszystkie właściwości komunikatu i pary nazwa-wartość pozostaną w pliku MQRFH2. Opcja ALL ma ten efekt tylko wtedy, gdy aplikacja odbierająca nie ustawiła opcji MQGMO_PROPERTIES lub ustawiła ją na wartość MQGMO_PROPERTIES_AS_Q_DEF.

Tabela 492. Ustawienia atrybutu właściwości komunikatu kolejki (kontynuacja)

PROPCTL	**Opis**
COMPAT (wartość domyślna)	Opcja COMPAT jest opcją domyślną. Jeśli parametr GMO_PROPERTIES_* nie jest ustawiony, tak jak w niezmodyfikowanej aplikacji z wcześniejszej wersji, przyjmowana jest wartość COMPAT . Ustawienie domyślne dla opcji COMPAT powoduje, że wcześniejsza wersja aplikacji, która nie utworzyła jawnie pliku MQRFH2, działa bez zmian w systemie IBM MQ 9.3. Tej opcji należy użyć, jeśli aplikacja MQI została napisana we wcześniejszej wersji, aby odczytywać komunikaty JMS . • Właściwości JMS , które są przechowywane w nagłówku MQRFH2 , są zwracane do aplikacji w nagłówku MQRFH2 w folderach o nazwach rozpoczynających się od mcd., jms., usr.lub mqext. • Jeśli komunikat zawiera foldery JMS i jeśli aplikacja IBM MQ 9.3 doda do komunikatu nowe foldery właściwości, te właściwości zostaną również zwrócone w pliku MQRFH2. W związku z tym należy zmodyfikować aplikację tak, aby obsługiwała nowe właściwości i nowe atrybuty właściwości. Zmiany w układzie i liczbie nagłówków MQRFH2 mogą mieć wpływ na niezmodyfikowaną aplikację. Niektóre atrybuty folderu mogą zostać usunięte lub program IBM MQ znajdzie błędy w układzie nagłówka MQRFH2 , który zignorował we wcześniejszej wersji. **Uwaga:** W tym scenariuszu działanie aplikacji jest takie samo, niezależnie od tego, czy jest ona połączona z wcześniejszą wersją, czy menedżerem kolejek produktu IBM MQ 9.3 . Jeśli atrybut kanału **PROPCTL** jest ustawiony na wartość COMPAT lub ALL , wszystkie nowe właściwości komunikatu są wysyłane w komunikacie do partnerskiego menedżera kolejek wcześniejszej wersji. • Jeśli komunikat nie jest komunikatem JMS , ale zawiera inne właściwości, te właściwości nie są zwracane do aplikacji w nagłówku MQRFH2 .[1] • Ta opcja umożliwia również poprawne działanie aplikacji we wcześniejszych wersjach, które w wielu przypadkach jawnie utworzyły MQRFH2 . Na przykład program MQI, który tworzy obiekt MQRFH2 zawierający właściwości komunikatu JMS , nadal działa poprawnie. Jeśli komunikat jest tworzony bez właściwości komunikatu JMS , ale z innymi folderami MQRFH2 , foldery są zwracane do aplikacji. Tylko wtedy, gdy foldery są folderami właściwości komunikatów, te konkretne foldery są usuwane z MQRFH2. Foldery właściwości komunikatów są identyfikowane przez dodanie nowego atrybutu folderu content='properties' lub są folderami o nazwach wymienionych w polu Nazwa zdefiniowanego folderu właściwości lub Niezgrupowana nazwa folderu właściwości. • Jeśli aplikacja wywołuje funkcję MQCRTMH w celu utworzenia uchwytu komunikatu, musi wysłać zapytanie do właściwości komunikatu przy użyciu funkcji MQINQMP. Właściwości komunikatu są usuwane z nagłówków MQRFH2 . Pary nazwa-wartość, które nie są właściwościami komunikatu, pozostają w MQRFH2. • Jeśli aplikacja wywołuje funkcję MQCRTMH w celu utworzenia uchwytu komunikatu, może wysłać zapytanie do wszystkich właściwości komunikatu, niezależnie od tego, czy komunikat zawiera foldery JMS . • Jeśli aplikacja nie utworzy uchwytu komunikatu, wszystkie właściwości komunikatu i pary nazwa-wartość pozostaną w pliku MQRFH2. Jeśli komunikat zawiera nowe foldery właściwości użytkownika, można wnioskować, że komunikat został utworzony przez nową lub zmienioną aplikację IBM MQ 9.3 . Jeśli aplikacja odbierająca ma przetwarzać te nowe właściwości bezpośrednio w MQRFH2, należy zmodyfikować aplikację tak, aby używała opcji ALL . W przypadku domyślnego zestawu opcji COMPAT niezmodyfikowana aplikacja kontynuuje przetwarzanie pozostałej części pliku MQRFH2bez właściwości IBM MQ 9.3 . Celem interfejsu PROPCTL jest obsługa starych aplikacji czytających foldery MQRFH2 oraz nowych i zmienionych aplikacji korzystających z interfejsu właściwości komunikatu. Należy dążyć do tego, aby nowe aplikacje używały interfejsu właściwości komunikatu dla wszystkich właściwości komunikatu użytkownika oraz aby uniknąć bezpośredniego odczytywania i zapisywania nagłówków MQRFH2 .

Tabela 492. Ustawienia atrybutu właściwości komunikatu kolejki (kontynuacja)	
PROPCTL	**Opis**
Wymuszenie	Opcja FORCE powoduje umieszczenie wszystkich właściwości komunikatów w nagłówkach MQRFH2 . Wszystkie właściwości komunikatu i pary nazwa-wartość w nagłówkach MQRFH2 pozostają w komunikacie. Właściwości komunikatu nie są usuwane z pliku MQRFH2i udostępniane za pośrednictwem uchwytu komunikatu. Efektem wybrania opcji FORCE jest umożliwienie nowo zmigrowanej aplikacji odczytywania właściwości komunikatu z nagłówków MQRFH2 . Załóżmy, że aplikacja została zmodyfikowana w celu przetwarzania właściwości komunikatu IBM MQ 9.3 , ale zachowała również możliwość bezpośredniej pracy z nagłówkami MQRFH2 , tak jak wcześniej. Użytkownik może zdecydować, kiedy przełączyć aplikację na używanie właściwości komunikatu, początkowo ustawiając atrybut kolejki PROPCTL na wartość FORCE. Ustaw atrybut kolejki **PROPCTL** na inną wartość, jeśli można rozpocząć korzystanie z właściwości komunikatu. Jeśli nowa funkcja w aplikacji nie działa zgodnie z oczekiwaniami, należy ustawić opcję **PROPCTL** z powrotem na wartość FORCE. Opcja FORCE ma ten efekt tylko wtedy, gdy aplikacja odbierająca nie ustawiła opcji MQGMO_PROPERTIES lub ustawiła ją na wartość MQGMO_PROPERTIES_AS_Q_DEF.
Brak	Należy użyć opcji NONE , aby istniejąca aplikacja mogła przetwarzać komunikat, ignorując wszystkie właściwości komunikatu, a nowa lub zmieniona aplikacja może tworzyć zapytania dotyczące właściwości komunikatu. • Jeśli aplikacja wywołuje funkcję MQCRTMH w celu utworzenia uchwytu komunikatu, musi wysłać zapytanie do właściwości komunikatu przy użyciu funkcji MQINQMP. Pary nazwa-wartość, które nie są właściwościami komunikatu, pozostają w pliku MQRFH2, który jest pozbawiony właściwości komunikatu. • Jeśli aplikacja nie utworzy uchwytu komunikatu, wszystkie właściwości komunikatu zostaną usunięte z pliku MQRFH2. Pary nazwa-wartość w nagłówkach MQRFH2 pozostają w komunikacie. Opcja NONE ma ten efekt tylko wtedy, gdy aplikacja odbierająca nie ustawiła opcji MQGMO_PROPERTIES lub ustawiła ją na wartość MQGMO_PROPERTIES_AS_Q_DEF.
V6COMPAT	Użyj tej opcji, aby otrzymać MQRFH2 w takim samym formacie, w jakim został wysłany. Jeśli aplikacja wysyłająca lub menedżer kolejek utworzy dodatkowe właściwości komunikatu, zostaną one zwrócone w uchwycie komunikatu. Ta opcja musi być ustawiona zarówno w kolejkach nadawczych, odbiorczych, jak i w pozostałych kolejkach transmisji. Nadpisuje wszystkie inne opcje PROPCTL ustawione w definicjach kolejek w ścieżce rozstrzygania nazw kolejek. Opcji V6COMPAT należy używać tylko w wyjątkowych okolicznościach. Jeśli na przykład przeprowadzana jest migracja aplikacji z wcześniejszej wersji do wersji IBM MQ 9.3, opcja ta jest przydatna, ponieważ zachowuje zachowanie wcześniejszej wersji. Ta opcja może mieć wpływ na przepustowość komunikatów. Administrowanie jest również trudniejsze; należy upewnić się, że ta opcja jest ustawiona w kolejkach nadawczych, odbiorczych i kolejkach transmisji. V6COMPAT ma ten efekt tylko wtedy, gdy aplikacja odbierająca nie ustawiła opcji MQGMO_PROPERTIES lub ustawiła ją na wartość MQGMO_PROPERTIES_AS_Q_DEF.

Więcej informacji na temat właściwości komunikatu i par nazwa-wartość zawiera sekcja “NameValueDane (MQCHARn) dla MQRFH2” na stronie 549.

[1] Istnienie konkretnych folderów właściwości utworzonych przez IBM MQ classes for JMS wskazuje komunikat JMS . Foldery właściwości to mcd., jms., usr.lub mqext.

Opcje właściwości komunikatu dla MQGMO

Opcje właściwości komunikatu **MQGMO** służą do sterowania sposobem zwracania właściwości komunikatu do aplikacji.

Tabela 493. Ustawienia opcji właściwości komunikatu MQGMO

MQGMO Opcja	**Opis**
MQGMO_PROPERTIES_AS_Q_DEF	Aplikacje IBM MQ , które odczytują dane z tej samej kolejki i nie ustawiają parametru GMO_PROPERTIES_*, odbierają właściwości komunikatu w inny sposób. Aplikacje IBM MQ , które nie tworzą uchwytu komunikatu, są sterowane przez atrybut **PROPCTL** kolejki. Aplikacja IBM MQ może odebrać właściwości komunikatu w MQRFH2lub utworzyć uchwyt komunikatu i wysłać zapytanie do właściwości komunikatu. Jeśli aplikacja tworzy uchwyt komunikatu, właściwości są usuwane z pliku MQRFH2. • Nowa lub zmieniona aplikacja IBM MQ , która nie ustawia parametru GMO_PROPERTIES_* lub nie ustawia go na wartość MQGMO_PROPERTIES_AS_Q_DEF , może wybrać zapytanie o właściwości komunikatu. Musi on ustawić właściwość MQCRTMH , aby utworzyć uchwyt komunikatu i właściwości komunikatu zapytania przy użyciu wywołania MQI produktu MQINQMP . • Jeśli nowa lub zmieniona aplikacja nie tworzy uchwytu komunikatu, musi odczytywać wszystkie właściwości komunikatu, które otrzymuje bezpośrednio z nagłówków MQRFH2 . • Jeśli atrybut kolejki **PROPCTL** jest ustawiony na wartość FORCE, w uchwycie komunikatu nie są zwracane żadne właściwości. Wszystkie właściwości są zwracane w nagłówkach MQRFH2 . • Jeśli atrybut kolejki **PROPCTL** jest ustawiony na wartość NONElub COMPAT, aplikacja IBM MQ , która tworzy uchwyt komunikatu, odbiera wszystkie właściwości komunikatu.
MQGMO_PROPERTIES_IN_HANDLE	Wymuś, aby aplikacja używała właściwości komunikatu. Użyj tej opcji, aby wykryć, czy zmodyfikowana aplikacja nie może utworzyć uchwytu komunikatu. Aplikacja może próbować odczytać właściwości komunikatu bezpośrednio z MQRFH2, zamiast wywoływać funkcję MQINQMP.
MQGMO_NO_PROPERTIES	• Wszystkie właściwości zostaną usunięte. Właściwości wygenerowane przez menedżer kolejek, takie jak właściwości JMS , zostaną usunięte. • Właściwości są usuwane nawet wtedy, gdy tworzony jest uchwyt komunikatu. Pary nazwa-wartość w innych folderach MQRFH2 są dostępne w danych komunikatu.

Tabela 493. Ustawienia opcji właściwości komunikatu MQGMO (kontynuacja)	
MQGMO Opcja	**Opis**
MQGMO_PROPERTIES_FORCE_MQRFH2	Właściwości są zwracane w nagłówkach MQRFH2 , nawet jeśli został utworzony uchwyt komunikatu. • Produkt MQINQMP nie zwraca żadnych właściwości komunikatu, nawet jeśli został utworzony uchwyt komunikatu. Wartość MQRC_PROPERTY_NOT_AVAILABLE jest zwracana, jeśli zostanie wykonane zapytanie o właściwość.
MQGMO_PROPERTIES_COMPATIBILITY	Jeśli komunikat pochodzi z klienta JMS , właściwości JMS są zwracane w nagłówkach MQRFH2 . Nowe lub zmodyfikowane aplikacje IBM MQ , które tworzą uchwyt komunikatu, zachowują się inaczej. • Wszystkie właściwości w dowolnym folderze właściwości komunikatu są zwracane, jeśli komunikat zawiera folder mcd., jms., usr.lub mqext . • Jeśli komunikat zawiera foldery właściwości, ale nie zawiera folderu mcd., jms., usr.lub mqext , w folderze MQRFH2nie są zwracane żadne właściwości komunikatu. • Jeśli nowa lub zmodyfikowana aplikacja IBM MQ tworzy uchwyt komunikatu, należy sprawdzić właściwości komunikatu przy użyciu wywołania MQI produktu MQINQMP . Wszystkie właściwości komunikatu są usuwane z pliku MQRFH2. • Jeśli nowa lub zmodyfikowana aplikacja IBM MQ tworzy uchwyt komunikatu, wszystkie właściwości w komunikacie mogą być odpytywane. Nawet jeśli komunikat nie zawiera folderu mcd., jms., usr.lub mqext , wszystkie właściwości komunikatu można przekształcić do postaci szeregowej.

Odsyłacze pokrewne
PROPCTL
2471 (09A7) (RC2471): WŁAŚCIWOŚĆ_MQRC_NIEDOSTĘPNA

StrucId (MQCHAR4) dla MQGMO

Jest to identyfikator struktury struktury opcji pobierania komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQGMO_STRUC_ID.

Wartość musi być następująca:

MQGMO_ID_struktury
Identyfikator struktury opcji pobierania komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQGMO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQGMO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQGMO

Wersja jest numerem wersji struktury.

Wartość musi być jedną z następujących wartości:

MQGMO_VERSION_1
Version-1 -struktura opcji get-message.

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQGMO_VERSION_2
Version-2 -struktura opcji get-message.

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQGMO_VERSION_3
Version-3 -struktura opcji get-message.

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQGMO_VERSION_4
Version-4 -struktura opcji get-message.

Ta wersja jest obsługiwana we wszystkich środowiskach.

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQGMO_CURRENT_VERSION
Bieżąca wersja struktury opcji pobierania komunikatów.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQGMO_VERSION_1.

Opcje (MQLONG) dla MQGMO

Opcje **MQGMO** sterują działaniem programu MQGET. Można podać zero lub więcej opcji. Jeśli potrzebna jest więcej niż jedna wartość opcjonalna:

- Dodaj wartości (nie dodawaj więcej niż raz tej samej stałej) lub
- Połącz wartości za pomocą bitowej operacji OR (jeśli język programowania obsługuje operacje bitowe).

Podano niepoprawne kombinacje opcji; wszystkie pozostałe kombinacje są poprawne.

Opcje oczekiwania

Następujące opcje odnoszą się do oczekiwania na przybycie komunikatów do kolejki:

MQGMO_WAIT
Aplikacja oczekuje na odpowiedni komunikat. Maksymalny czas oczekiwania aplikacji jest określony w pliku *WaitInterval*.

Ważne: Nie ma oczekiwania ani opóźnienia, jeśli odpowiedni komunikat jest dostępny natychmiast.

Jeśli żądania MQGET są zablokowane lub jeśli żądania MQGET zostaną zablokowane podczas oczekiwania, oczekiwanie zostanie anulowane. Wywołanie kończy się z kodem MQCC_FAILED i kodem przyczyny MQRC_GET_INHIBITED, niezależnie od tego, czy w kolejce znajdują się odpowiednie komunikaty.

Opcji MQGMO_WAIT można używać z opcjami MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT .

Jeśli kilka aplikacji oczekuje w tej samej kolejce współużytkowanej, następujące reguły określają, która aplikacja jest aktywowana po nadejściu odpowiedniego komunikatu:

Tabela 494. Reguły aktywowania wywołań MQGET w kolejce współużytkowanej.

Liczba wywołań MQGET oczekujących na aktywację		**Wynik**
Z opcją BROWSE	**Bez opcji BROWSE**[2]	
Brak	jeden lub więcej	Aktywowane jest jedno wywołanie MQGET bez opcji BROWSE .
jeden lub więcej	Brak	Aktywowane są wszystkie wywołania MQGET z opcją BROWSE .

[2] Wywołanie MQGET z opcją MQGMO_LOCK jest traktowane jako wywołanie bez przeglądania.

Tabela 494. Reguły aktywowania wywołań MQGET w kolejce współużytkowanej. (kontynuacja)		
Liczba wywołań MQGET oczekujących na aktywację		**Wynik**
Z opcją BROWSE	**Bez opcji BROWSE[2]**	
jeden lub więcej	jeden lub więcej	Aktywowane jest jedno wywołanie MQGET bez opcji BROWSE . Liczba aktywnych wywołań MQGET z opcją BROWSE jest nieprzewidywalna.

Jeśli więcej niż jedno wywołanie MQGET bez opcji BROWSE oczekuje w tej samej kolejce, aktywowane jest tylko jedno. Menedżer kolejek próbuje nadać priorytet oczekującym wywołaom w następującej kolejności:

1. Konkretne żądania oczekiwania na pobranie, które mogą być spełnione tylko przez określone komunikaty, na przykład te z konkretnym MsgId lub CorrelId (lub oba te komunikaty).
2. Ogólne żądania oczekiwania na pobranie, które mogą być spełnione przez dowolny komunikat.

Uwaga:

- W pierwszej kategorii nie jest nadawany dodatkowy priorytet dla bardziej konkretnych żądań oczekiwania na pobranie. Na przykład żądania, które określają zarówno MsgId , jak i CorrelId .
- W żadnej z tych kategorii nie można przewidzieć, która aplikacja jest wybrana. W szczególności aplikacja oczekująca najdłużej nie musi być tą wybraną.
- Długość ścieżki i uwagi dotyczące planowania priorytetu w systemie operacyjnym mogą oznaczać, że oczekująca aplikacja o niższym priorytecie systemu operacyjnego niż oczekiwano pobiera komunikat.
- Może się również zdarzyć, że aplikacja, która nie oczekuje, pobierze komunikat w preferowanej kolejności.

z/OS W systemie z/OSmają zastosowanie następujące punkty:

- Jeśli aplikacja ma kontynuować pracę podczas oczekiwania na nadejście komunikatu, należy rozważyć użycie opcji sygnału (MQGMO_SET_SIGNAL). Jednak opcja sygnału jest specyfika środowiska; aplikacje, które mają być używane między różnymi środowiskami, nie mogą z niej korzystać.
- Jeśli istnieje więcej niż jedno wywołanie MQGET oczekujące na ten sam komunikat, z kombinacją opcji oczekiwania i sygnału, każde wywołanie oczekujące jest traktowane jako równe. Podanie wartości MQGMO_SET_SIGNAL z wartością MQGMO_WAITjest błędem. Określenie tej opcji z uchwytem kolejki, dla którego sygnał jest oczekujący, jest również błędem.
- W przypadku określenia wartości MQGMO_WAIT lub MQGMO_SET_SIGNAL dla kolejki, która ma IndexType o wartości MQIT_MSG_TOKEN, nie są dozwolone żadne kryteria wyboru. Oznacza to, że:
  - Jeśli używana jest wersja version-1 MQGMO, ustaw pola MsgId i CorrelId w polu MQMD określonym w wywołaniu MQGET na wartości MQMI_NONE i MQCI_NONE.
  - Jeśli używana jest wersja version-2 lub nowsza MQGMO, w polu MatchOptions należy ustawić wartość MQMO_NONE.
- W przypadku wywołania MQGET w kolejce współużytkowanej, które jest żądaniem przeglądania lub destrukcyjnym pobraniem komunikatu grupy, a nie ma potrzeby dopasowania wartości MsgId ani CorrelId , sygnał EBC jest wysyłany po upływie 200 milisekund.

  Taka sytuacja ma miejsce, nawet jeśli odpowiedni komunikat nie dotarł do kolejki, dopóki nie upłynie odstęp czasu oczekiwania, gdy kolejka zostanie wysłana z wartością

[2] Wywołanie MQGET z opcją MQGMO_LOCK jest traktowane jako wywołanie bez przeglądania.
[2] Wywołanie MQGET z opcją MQGMO_LOCK jest traktowane jako wywołanie bez przeglądania.

MQEC_WAIT_INTERVAL_EXPIRED. Po wysłaniu komunikatu MQEC_MSG_NADESZŁA należy ponownie wywołać drugie wywołanie MQGET w celu pobrania komunikatu, jeśli jest on dostępny.

Ta technika jest używana w celu zapewnienia, że użytkownik zostanie poinformowany w odpowiednim czasie o nadejściu komunikatu, ale może pojawić się jako nieoczekiwany narzut przetwarzania w porównaniu z podobną sekwencją wywołań w niewspółużytkowanej kolejce.

Parametr MQGMO_WAIT jest ignorowany, jeśli zostanie podany razem z parametrem MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_MSG_UNDER_CURSOR ; nie został zgłoszony żaden błąd.

MQGMO_NO_WAIT
Aplikacja nie czeka, jeśli nie jest dostępny odpowiedni komunikat. MQGMO_NO_WAIT jest przeciwieństwem MQGMO_WAIT. Zmienna MQGMO_NO_WAIT jest zdefiniowana jako pomocna w dokumentacji programu. Jest to wartość domyślna, jeśli nie określono żadnej z nich.

MQGMO_SET_SIGNAL
Tej opcji należy używać w połączeniu z polami Signal1 i Signal2 . Umożliwia ona aplikacjom kontynuowanie pracy podczas oczekiwania na nadejście komunikatu. Umożliwia również aplikacjom (jeśli dostępne są odpowiednie narzędzia systemu operacyjnego) oczekiwanie na komunikaty przychodzące do więcej niż jednej kolejki.

Uwaga: Opcja MQGMO_SET_SIGNAL jest specyfika dla środowiska; nie należy jej używać dla aplikacji, które mają być używane jako porty.

W dwóch okolicznościach wywołanie kończy się w taki sam sposób, jak gdyby opcja ta nie została określona:

1. Jeśli aktualnie dostępny komunikat spełnia kryteria określone w deskryptorze komunikatu.
2. Jeśli zostanie wykryty błąd parametru lub inny błąd synchroniczny.

Jeśli żaden komunikat spełniający kryteria określone w deskryptorze komunikatu nie jest obecnie dostępny, sterowanie wraca do aplikacji bez oczekiwania na odebranie komunikatu. Parametry **CompCode** i **Reason** są ustawione na wartości MQCC_WARNING i MQRC_SIGNAL_REQUEST_ACCEPTED. Inne pola wyjściowe w deskryptorze komunikatu i parametry wyjściowe wywołania MQGET nie są ustawione. Po otrzymaniu odpowiedniego komunikatu w późniejszym terminie, sygnał jest wysyłany przez EBC.

Następnie program wywołujący musi ponownie wywołać funkcję MQGET , aby pobrać komunikat. Aplikacja może oczekiwać na ten sygnał, używając funkcji udostępnianych przez system operacyjny.

Jeśli system operacyjny udostępnia mechanizm wielokrotnego oczekiwania, można go użyć do oczekiwania na komunikat nadchodzący do dowolnej z kilku kolejek.

Jeśli zostanie podana wartość WaitInterval różna od zera, sygnał jest dostarczany po upływie czasu oczekiwania. Menedżer kolejek może również anulować oczekiwanie, w którym to przypadku sygnał jest dostarczany.

Więcej niż jedno wywołanie MQGET może ustawić sygnał dla tego samego komunikatu. Kolejność aktywowania aplikacji jest taka sama, jak opisana w sekcji MQGMO_WAIT.

Jeśli więcej niż jedno wywołanie MQGET oczekuje na ten sam komunikat, każde oczekujące wywołanie jest traktowane jednakowo. Wywołania mogą zawierać kombinację opcji oczekiwania i sygnału.

W pewnych warunkach wywołanie funkcji MQGET może pobrać komunikat, a sygnał otrzymany w wyniku nadejścia tego samego komunikatu może zostać dostarczony. Po dostarczeniu sygnału należy przygotować aplikację, aby komunikat nie był dostępny.

Uchwyt kolejki może mieć nie więcej niż jedno oczekujące żądanie sygnału.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_UNLOCK
- MQGMO_WAIT

W przypadku wywołania MQGET w kolejce współużytkowanej, które jest żądaniem przeglądania lub destrukcyjnym pobraniem komunikatu grupy, a nie ma potrzeby dopasowania wartości MsgId ani CorrelId , sygnał użytkownika EBC jest wysyłany MQEC_MSG_ARRIVED po upływie 200 milisekund.

Taka sytuacja występuje, nawet jeśli odpowiedni komunikat nie dotarł do kolejki, dopóki nie upłynie czas oczekiwania, gdy kolejka zostanie wysłana za pomocą komendy MQEC_WAIT_INTERVAL_EXPIRED. Po opublikowaniu komunikatu MQEC_MSG_ARRIVED konieczne jest ponowne wywołanie metody MQGET w celu pobrania komunikatu, jeśli jest on dostępny.

Ta technika jest używana w celu zapewnienia, że użytkownik zostanie poinformowany w odpowiednim czasie o nadejściu komunikatu, ale może pojawić się jako nieoczekiwany narzut przetwarzania w porównaniu z podobną sekwencją wywołań w niewspółużytkowanej kolejce.

Nie jest to wydajna metoda pobierania komunikatów, gdy komunikaty są dodawane rzadko. Aby uniknąć tego narzutu w przypadku przeglądania, należy podać parametr MsgId (jeśli nie jest indeksowany lub indeksowany przez MsgId) lub *CorrelId* (jeśli jest indeksowany przez CorrelId) w wywołaniu funkcji MQGET .

z/OS Ta opcja jest obsługiwana tylko w systemie z/OS .

MQGMO_FAIL_IF_QUIESCING,
Wymuś niepowodzenie wywołania MQGET , jeśli menedżer kolejek jest w stanie wyciszania.

z/OS W systemie z/OSta opcja wymusza również niepowodzenie wywołania MQGET , jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wyciszenia.

Jeśli ta opcja jest określona razem z opcją MQGMO_WAIT lub MQGMO_SET_SIGNAL, a oczekiwanie lub sygnał jest oczekujący w momencie, gdy menedżer kolejek przechodzi w stan wyciszania:

- Oczekiwanie zostało anulowane, a wywołanie zwraca kod zakończenia MQCC_FAILED z kodem przyczyny MQRC_Q_MGR_QUIESCING lub MQRC_CONNECTION_QUIESCING.
- Sygnał jest anulowany z kodem zakończenia sygnału specyficznego dla środowiska.

  z/OS W systemie z/OSsygnał kończy się kodem zakończenia zdarzenia MQEC_Q_MGR_QUIESCING lub MQEC_CONNECTION_QUIESCING.

Jeśli parametr MQGMO_FAIL_IF_QUIESCING nie jest określony, a menedżer kolejek lub połączenie przechodzi w stan wyciszania, oczekiwanie lub sygnał nie jest anulowany.

Opcje punktu synchronizacji

Następujące opcje odnoszą się do uczestnictwa w wywołaniu programu MQGET w jednostce pracy:

MQGMO_SYNCPOINT
Żądanie ma działać w ramach zwykłych protokołów jednostki pracy. Komunikat jest oznaczany jako niedostępny dla innych aplikacji, ale jest usuwany z kolejki tylko wtedy, gdy jednostka pracy jest zatwierdzona. Komunikat zostanie ponownie udostępniony, jeśli jednostka pracy zostanie wycofana.

Można pozostawić nieustawione opcje MQGMO_SYNCPOINT i MQGMO_NO_SYNCPOINT . W takim przypadku uwzględnienie żądania get w protokołach jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek. Nie jest ona określana przez środowisko, w którym działa aplikacja.

- z/OS W systemie z/OSżądanie get znajduje się w jednostce pracy.
- We wszystkich środowiskach z wyjątkiem środowiska z/OSżądanie pobrania nie znajduje się w jednostce pracy.

Z powodu tych różnic aplikacja, która ma być używana jako port, nie może zezwalać na ustawienie tej opcji jako domyślnej; należy jawnie określić wartość MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT .

Ta opcja nie jest poprawna z żadną z następujących opcji:

- 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,
Żądanie ma działać w ramach zwykłych protokołów jednostki pracy, ale tylko wtedy, gdy pobierany komunikat jest trwały. Komunikat trwały ma wartość MQPER_PERSISTENT w polu Persistence w strukturze MQMD.

- Jeśli komunikat jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby w aplikacji określono parametr MQGMO_SYNCPOINT.
- Jeśli komunikat nie jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby w aplikacji określono parametr MQGMO_NO_SYNCPOINT.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- 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

Ta opcja jest obsługiwana w następujących środowiskach:

- AIX
- IBM i
- Linux
- z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

MQGMO_NO_SYNCPOINT
Żądanie ma działać poza normalnymi protokołami jednostki pracy. Jeśli zostanie wyświetlony komunikat bez opcji przeglądania, zostanie on natychmiast usunięty z kolejki. Komunikat nie może zostać ponownie udostępniony przez wycofanie jednostki pracy.

Ta opcja jest używana, jeśli zostanie podana opcja MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

Można pozostawić nieustawione opcje MQGMO_SYNCPOINT i MQGMO_NO_SYNCPOINT . W takim przypadku uwzględnienie żądania get w protokołach jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek. Nie jest ona określana przez środowisko, w którym działa aplikacja.

- z/OS W systemie z/OSżądanie get znajduje się w jednostce pracy.
- We wszystkich środowiskach z wyjątkiem środowiska z/OSżądanie pobrania nie znajduje się w jednostce pracy.

Z powodu tych różnic aplikacja, która ma być używana jako port, nie może zezwalać na ustawienie tej opcji jako domyślnej. Należy jawnie określić wartość MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT .

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT

z/OS **MQGMO_MARK_SKIP_BACKOUT**

Wycofuje jednostkę pracy bez przywracania do kolejki komunikatu oznaczonego tą opcją.

Ta opcja jest obsługiwana tylko w systemie z/OS.

Jeśli ta opcja jest określona, należy również podać parametr MQGMO_SYNCPOINT . Wartość MQGMO_MARK_SKIP_BACKOUT nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_LOCK
- MQGMO_NO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Uwaga: W systemach IMS i CICSmoże być konieczne wywołanie metody extran IBM MQ po wycofaniu jednostki pracy zawierającej komunikat z oznaczeniem MQGMO_MARK_SKIP_BACKOUT. Przed zatwierdzeniem nowej jednostki pracy zawierającej oznaczony komunikat należy wywołać funkcję IBM MQ . Może to być dowolne połączenie IBM MQ .

1. W systemie IMS, jeśli nie zastosowano raportu IMS APAR PN60855 i uruchomiona jest aplikacja IMS MPP lub BMP.
2. W systemie CICS, jeśli uruchomiona jest dowolna aplikacja.

W obu przypadkach należy wywołać funkcję IBM MQ przed zatwierdzeniem nowej jednostki pracy zawierającej wycofany komunikat.

Uwaga: W ramach jednostki pracy może istnieć tylko jedno żądanie pobrania oznaczone jako pomijane wycofanie, a także jedno lub kilka nieoznaczonych żądań pobrania.

Jeśli aplikacja wycofa się z jednostki pracy, komunikat pobrany za pomocą komendy MQGMO_MARK_SKIP_BACKOUT nie zostanie odtworzony do poprzedniego stanu. Inne aktualizacje zasobów są wycofywane. Komunikat jest traktowany tak, jakby został pobrany w nowej jednostce pracy uruchomionej przez żądanie wycofania. Komunikat jest pobierany bez opcji MQGMO_MARK_SKIP_BACKOUT .

Parametr MQGMO_MARK_SKIP_BACKOUT jest użyteczny, gdy po zmianie niektórych zasobów okazuje się, że jednostka pracy nie może zostać pomyślnie zakończona. Jeśli ta opcja zostanie pominięta, wycofanie jednostki pracy spowoduje przywrócenie komunikatu w kolejce. Ta sama sekwencja zdarzeń występuje ponownie, gdy komunikat jest pobierany jako następny.

Jeśli jednak w oryginalnym wywołaniu funkcji MQGET zostanie podana wartość MQGMO_MARK_SKIP_BACKOUT , wycofanie jednostki pracy spowoduje wycofanie aktualizacji innych zasobów. Komunikat jest traktowany tak, jakby został pobrany w ramach nowej jednostki pracy. Aplikacja może wykonać odpowiednią obsługę błędów. Może on wysłać komunikat raportu do nadawcy oryginalnego komunikatu lub umieścić oryginalny komunikat w kolejce niedostarczonych komunikatów. Następnie może zatwierdzić nową jednostkę pracy. Zatwierdzenie nowej jednostki pracy powoduje trwałe usunięcie komunikatu z oryginalnej kolejki.

MQGMO_MARK_SKIP_BACKOUT oznacza pojedynczy komunikat fizyczny. Jeśli komunikat należy do grupy komunikatów, pozostałe komunikaty w grupie nie są oznaczane. Podobnie, jeśli oznaczony komunikat jest segmentem komunikatu logicznego, pozostałe segmenty komunikatu logicznego nie są oznaczone.

Każdy komunikat w grupie może zostać oznaczony, ale jeśli komunikaty są pobierane za pomocą programu MQGMO_LOGICAL_ORDER, korzystne jest oznaczenie pierwszego komunikatu w grupie.

Jeśli jednostka pracy zostanie wycofana, pierwszy (oznaczony) komunikat zostanie przeniesiony do nowej jednostki pracy. Drugi i późniejszy komunikat w grupie są przywracane do kolejki. Komunikaty pozostawione w kolejce nie mogą być pobierane przez inną aplikację za pomocą programu MQGMO_LOGICAL_ORDER. Pierwszy komunikat w grupie nie znajduje się już w kolejce. Jednak aplikacja, która oparła jednostkę pracy, może pobrać drugi i późniejsze komunikaty do nowej jednostki pracy za pomocą opcji MQGMO_LOGICAL_ORDER . Pierwszy komunikat został już pobrany.

Czasami konieczne może być wycofanie nowej jednostki pracy. Na przykład, ponieważ kolejka niedostarczonych komunikatów jest pełna i komunikat nie może zostać odrzucony. Wycofanie nowej jednostki pracy powoduje przywrócenie komunikatu w oryginalnej kolejce, co zapobiega utracie komunikatu. Jednak w tej sytuacji przetwarzanie nie może być kontynuowane. Po wycofaniu nowej jednostki pracy aplikacja musi poinformować operatora lub administratora, że wystąpił nienaprawialny błąd, a następnie zakończyć działanie.

Funkcja MQGMO_MARK_SKIP_BACKOUT działa tylko wtedy, gdy jednostka pracy zawierająca żądanie get została przerwana przez aplikację, która ją wycofała. Jeśli jednostka pracy zawierająca żądanie pobrania została wycofana z powodu niepowodzenia transakcji lub systemu, wartość MQGMO_MARK_SKIP_BACKOUT jest ignorowana. Wszystkie komunikaty pobrane za pomocą tej opcji są przywracane do kolejki w taki sam sposób, jak komunikaty pobrane bez tej opcji.

Przeglądaj opcje

Następujące opcje odnoszą się do przeglądania komunikatów w kolejce:

MQGMO_BROWSE_FIRST
Gdy kolejka jest otwierana z opcją MQOO_BROWSE, ustanawiany jest kursor przeglądania, umieszczony logicznie przed pierwszym komunikatem w kolejce. Następnie można użyć wywołań MQGET , określając opcję MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXTlub MQGMO_BROWSE_MSG_UNDER_CURSOR w celu pobrania komunikatów z kolejki bez zniszczenia. Kursor przeglądania oznacza pozycję w obrębie komunikatów w kolejce, od której następne wywołanie MQGET z MQGMO_BROWSE_NEXT wyszukuje odpowiedni komunikat.

Wartość MQGMO_BROWSE_FIRST nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

Wywołanie funkcji MQGET z opcją MQGMO_BROWSE_FIRST ignoruje poprzednią pozycję kursora przeglądania. Pobierany jest pierwszy komunikat w kolejce, który spełnia warunki określone w deskryptorze komunikatu. Komunikat pozostaje w kolejce, a kursor przeglądania jest ustawiony na tym komunikacie.

Po tym wywołaniu kursor przeglądania jest umieszczany na zwróconym komunikacie. Komunikat może zostać usunięty z kolejki przed następnym wywołaniem metody MQGET z wartością MQGMO_BROWSE_NEXT . W tym przypadku kursor przeglądania pozostaje na pozycji w kolejce zajmowanej przez komunikat, nawet jeśli ta pozycja jest teraz pusta.

Aby usunąć komunikat z kolejki, należy użyć opcji MQGMO_MSG_UNDER_CURSOR z wywołaniem MQGET bez przeglądania.

Kursor przeglądania nie jest przenoszony przez wywołanie funkcji MQGET bez przeglądania, nawet jeśli używany jest ten sam uchwyt *Hobj* . Nie jest też przenoszona przez wywołanie przeglądania MQGET , które zwraca kod zakończenia MQCC_FAILEDlub kod przyczyny MQRC_TRUNCATED_MSG_FAILED.

Aby zablokować przeglądany komunikat, należy razem z tą opcją podać opcję MQGMO_LOCK .

Parametr MQGMO_BROWSE_FIRST można podać z dowolną poprawną kombinacją opcji MQGMO_* i MQMO_* , które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli zostanie podana wartość MQGMO_LOGICAL_ORDER, komunikaty będą przeglądane w kolejności logicznej. Jeśli ta opcja zostanie pominięta, komunikaty będą przeglądane w kolejności fizycznej. Jeśli zostanie podana wartość MQGMO_BROWSE_FIRST, można przełączać się między kolejnością logiczną a kolejnością fizyczną. Kolejne wywołania MQGET używające MQGMO_BROWSE_NEXT przeglądają kolejkę w tej samej kolejności, co ostatnie wywołanie, w którym określono MQGMO_BROWSE_FIRST dla uchwytu kolejki.

Menedżer kolejek przechowuje dwa zestawy informacji o grupach i segmentach dla wywołań MQGET . Informacje o grupie i segmencie dla wywołań przeglądania są zachowywane oddzielnie od informacji dla wywołań, które usuwają komunikaty z kolejki. Jeśli zostanie podana wartość MQGMO_BROWSE_FIRST, menedżer kolejek ignoruje informacje o grupie i segmencie na potrzeby przeglądania. Skanuje kolejkę tak, jakby nie było bieżącej grupy ani bieżącego komunikatu logicznego. Jeśli wywołanie MQGET zakończyło się pomyślnie, kod zakończenia MQCC_OK lub MQCC_WARNING, informacje o grupie i segmencie do przeglądania są ustawiane na informacje o zwróconym komunikacie. Jeśli wywołanie nie powiedzie się, informacje o grupie i segmencie pozostają takie same, jak przed wywołaniem.

MQGMO_BROWSE_NEXT
Przesuń kursor przeglądania do następnego komunikatu w kolejce, który spełnia kryteria wyboru określone w wywołaniu funkcji MQGET . Komunikat jest zwracany do aplikacji, ale pozostaje w kolejce.

Wartość MQGMO_BROWSE_NEXT nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

Działanie programu MQGMO_BROWSE_NEXT jest takie samo jak działanie programu MQGMO_BROWSE_FIRST, jeśli jest to pierwsze wywołanie przeglądania kolejki po otwarciu kolejki do przeglądania.

Komunikat znajdujący się pod kursorem może zostać usunięty z kolejki przed wysłaniem następnego wywołania MQGET z MQGMO_BROWSE_NEXT . Kursor przeglądania pozostaje logicznie na pozycji w kolejce zajmowanej przez komunikat, nawet jeśli pozycja ta jest teraz pusta.

Komunikaty są przechowywane w kolejce na jeden z dwóch sposobów:

- FIFO w ramach priorytetu (MQMDS_PRIORITY) lub
- FIFO niezależnie od priorytetu (MQMDS_FIFO)

Atrybut kolejki **MsgDeliverySequence** wskazuje, która metoda ma zastosowanie (szczegółowe informacje zawiera sekcja “Atrybuty kolejek” na stronie 863).

Kolejka może mieć MsgDeliverySequence MQMDS_PRIORITY. W kolejce pojawia się komunikat o wyższym priorytecie niż ten, który jest obecnie wskazywany przez kursor przeglądania. W takim przypadku komunikat o wyższym priorytecie nie zostanie znaleziony podczas bieżącego przeglądania kolejki za pomocą programu MQGMO_BROWSE_NEXT. Można go znaleźć tylko po zresetowaniu kursora przeglądania za pomocą wartości MQGMO_BROWSE_FIRSTlub po ponownym otwarciu kolejki.

W razie potrzeby można użyć opcji MQGMO_MSG_UNDER_CURSOR z wywołaniem MQGET bez przeglądania, aby usunąć komunikat z kolejki.

Kursor przeglądania nie jest przenoszony przez wywołania funkcji MQGET bez przeglądania przy użyciu tego samego uchwytu Hobj .

Razem z tą opcją należy podać opcję MQGMO_LOCK , aby zablokować przeglądany komunikat.

Parametr MQGMO_BROWSE_NEXT można podać z dowolną poprawną kombinacją opcji MQGMO_* i MQMO_* , które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli zostanie podana wartość MQGMO_LOGICAL_ORDER, komunikaty będą przeglądane w kolejności logicznej. Jeśli ta opcja zostanie pominięta, komunikaty będą przeglądane w kolejności fizycznej. Jeśli zostanie podana wartość MQGMO_BROWSE_FIRST, można przełączać się między kolejnością logiczną a kolejnością fizyczną. Kolejne wywołania MQGET używające MQGMO_BROWSE_NEXT przeglądają kolejkę w tej samej kolejności, co ostatnie wywołanie, w którym określono MQGMO_BROWSE_FIRST dla uchwytu kolejki. Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_INCONSISTENT_BROWSE , jeśli ten warunek nie jest spełniony.

Uwaga: Należy zachować szczególną ostrożność podczas korzystania z wywołania MQGET w celu przejścia poza koniec grupy komunikatów, jeśli nie określono parametru MQGMO_LOGICAL_ORDER . Załóżmy na przykład, że ostatni komunikat w grupie poprzedza pierwszy komunikat w grupie w kolejce. Użycie opcji MQGMO_BROWSE_NEXT do przeglądania poza końcem grupy, podanie opcji MQMO_MATCH_MSG_SEQ_NUMBER z wartością MsgSeqNumber ustawioną na 1 spowoduje zwrócenie pierwszego komunikatu w grupie, która była już przeglądnięta. Ten wynik może wystąpić natychmiast lub kilka wywołań MQGET później, jeśli istnieją grupy pośredniczące. To samo dotyczy komunikatu logicznego, który nie znajduje się w grupie.

Informacje o grupie i segmencie dla wywołań przeglądania są zachowywane oddzielnie od informacji dla wywołań, które usuwają komunikaty z kolejki.

MQGMO_BROWSE_MSG_UNDER_CURSOR
Pobierz nieniszczący komunikat wskazywany przez kursor przeglądania, niezależnie od opcji MQMO_* określonych w polu MatchOptions w MQGMO.

Wartość MQGMO_BROWSE_MSG_UNDER_CURSOR nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_MSG_UNDER_CURSOR
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

Komunikat wskazywany przez kursor przeglądania jest ostatnim komunikatem pobranym przy użyciu opcji MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT . Wywołanie nie powiedzie się, jeśli żadne z tych wywołań nie zostało wykonane dla tej kolejki od czasu jej otwarcia. Wywołanie również kończy się niepowodzeniem, jeśli komunikat, który znajdował się pod kursorem przeglądania, został od tego czasu pobrany ze zniszczeniem.

Pozycja kursora przeglądania nie jest zmieniana przez to wywołanie.

Aby usunąć komunikat z kolejki, można użyć opcji MQGMO_MSG_UNDER_CURSOR z wywołaniem MQGET bez przeglądania.

Kursor przeglądania nie jest przenoszony przez wywołanie funkcji MQGET bez przeglądania, nawet jeśli używany jest ten sam uchwyt Hobj . Nie jest też przenoszona przez wywołanie przeglądania MQGET , które zwraca kod zakończenia MQCC_FAILEDlub kod przyczyny MQRC_TRUNCATED_MSG_FAILED.

Jeśli parametr MQGMO_BROWSE_MSG_UNDER_CURSOR jest określony razem z parametrem MQGMO_LOCK:

- Jeśli komunikat jest już zablokowany, musi to być ten, który znajduje się pod kursorem, aby został zwrócony bez ponownego odblokowywania i blokowania. Komunikat pozostaje zablokowany.
- Jeśli nie ma zablokowanego komunikatu i pod kursorem przeglądania znajduje się komunikat, jest on blokowany i zwracany do aplikacji. Jeśli pod kursorem przeglądania nie ma żadnego komunikatu, wywołanie nie powiedzie się.

Jeśli parametr MQGMO_BROWSE_MSG_UNDER_CURSOR jest określony bez parametru MQGMO_LOCK:

- Jeśli komunikat jest już zablokowany, musi to być ten, który znajduje się pod kursorem. Komunikat zostanie zwrócony do aplikacji, a następnie odblokowany. Ponieważ komunikat jest teraz odblokowany, nie ma gwarancji, że będzie można go ponownie przeglądać lub pobierać destrukcyjnie przez tę samą aplikację. Możliwe, że został on pobrany destrukcyjnie przez inną aplikację pobierając komunikaty z kolejki.
- Jeśli nie ma zablokowanego komunikatu i pod kursorem przeglądania znajduje się komunikat, jest on zwracany do aplikacji. Jeśli pod kursorem przeglądania nie ma żadnego komunikatu, wywołanie nie powiedzie się.

Jeśli parametr MQGMO_COMPLETE_MSG jest określony z parametrem MQGMO_BROWSE_MSG_UNDER_CURSOR, kursor przeglądania musi identyfikować komunikat, którego pole Offset w strukturze MQMD ma wartość zero. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_INVALID_MSG_UNDER_CURSOR.

Informacje o grupie i segmencie dla wywołań przeglądania są zachowywane oddzielnie od informacji dla wywołań, które usuwają komunikaty z kolejki.

MQGMO_MSG_UNDER_CURSOR
Pobierz komunikat wskazywany przez kursor przeglądania, niezależnie od opcji MQMO_* określonych w polu MatchOptions w produkcie MQGMO. Komunikat jest usuwany z kolejki.

Komunikat wskazywany przez kursor przeglądania jest ostatnim komunikatem pobranym przy użyciu opcji MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT .

Jeśli parametr MQGMO_COMPLETE_MSG jest określony z parametrem MQGMO_MSG_UNDER_CURSOR, kursor przeglądania musi identyfikować komunikat, którego pole Offset w strukturze MQMD ma wartość zero. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_INVALID_MSG_UNDER_CURSOR.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT
- MQGMO_UNLOCK

Jest to również błąd, jeśli kolejka nie została otwarta zarówno do przeglądania, jak i do wprowadzania. Jeśli kursor przeglądania nie wskazuje aktualnie na komunikat, który można pobrać, wywołanie MQGET zwraca błąd.

MQGMO_MARK_BROWSE_HANDLE
Komunikat, który został zwrócony przez pomyślną funkcję MQGETlub identyfikowany przez zwróconą funkcję MsgToken, jest oznaczany. Znacznik jest specyficzny dla uchwytu obiektu używanego w wywołaniu.

Komunikat nie jest usuwany z kolejki.

Parametr MQGMO_MARK_BROWSE_HANDLE jest poprawny tylko wtedy, gdy podano jedną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR

- MQGMO_BROWSE_NEXT

Wartość MQGMO_MARK_BROWSE_HANDLE nie jest poprawna z żadną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Komunikat pozostaje w tym stanie do momentu wystąpienia jednego z następujących zdarzeń:

- Uchwyt obiektu, którego to dotyczy, jest zamknięty normalnie lub w inny sposób.
- Komunikat jest oznaczany dla tego uchwytu przez wywołanie MQGET z opcją MQGMO_UNMARK_BROWSE_HANDLE.
- Komunikat jest zwracany z wywołania niszczycielskiego programu MQGET, które kończy się wartością MQCC_OK lub MQCC_WARNING. Stan komunikatu pozostaje zmieniony nawet wtedy, gdy MQGET zostanie później wycofany.
- Komunikat traci ważność.

MQGMO_MARK_BROWSE_CO_OP

Komunikat zwracany przez pomyślną funkcję MQGETlub identyfikowany przez zwróconą funkcję *MsgToken*jest oznaczany dla wszystkich uchwytów w zestawie współpracującym.

Znacznik poziomu kooperacji jest uzupełnieniem znacznika poziomu uchwytu, który mógł zostać ustawiony.

Komunikat nie jest usuwany z kolejki.

Parametr MQGMO_MARK_BROWSE_CO_OP jest poprawny tylko wtedy, gdy użyty uchwyt obiektu został zwrócony przez wywołanie do MQOPEN , które określo MQOO_CO_OP. Należy również określić jedną z następujących opcji MQGMO :

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_BROWSE_NEXT

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

Jeśli komunikat jest już oznaczony, a opcja MQGMO_UNMARKED_BROWSE_MSG nie jest określona, wywołanie kończy się niepowodzeniem z kodem MQCC_FAILED i kodem przyczyny MQRC_MSG_MARKED_BROWSE_CO_OP.

Komunikat pozostaje w tym stanie do momentu wystąpienia jednego z następujących zdarzeń:

- Wszystkie uchwyty obiektów w zestawie kooperującym są zamknięte.
- Komunikat nie jest oznaczany dla współpracujących przeglądarek za pomocą wywołania MQGET z opcją MQGMO_UNMARK_BROWSE_CO_OP.
- Menedżer kolejek automatycznie usuwa oznaczenie komunikatu.
- Komunikat jest zwracany z wywołania funkcji MQGET, która nie jest przeglądaniem. Stan komunikatu pozostaje zmieniony nawet wtedy, gdy MQGET zostanie później wycofany.

- Komunikat traci ważność.

MQGMO_UNMARKED_BROWSE_MSG
Wywołanie metody MQGET , która określa MQGMO_UNMARKED_BROWSE_MSG , zwraca komunikat, który jest traktowany jako nieoznaczony jako uchwyt. Komunikat nie jest zwracany, jeśli został oznaczony do obsługi. Komunikat nie jest zwracany również wtedy, gdy kolejka została otwarta przez wywołanie funkcji MQOPENz opcją MQOO_CO_OP, a komunikat został oznaczony przez element zestawu współpracującego.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOCK
- MQGMO_LOGICAL_ORDER
- MQGMO_UNLOCK

MQGMO_UNMARK_BROWSE_CO_OP
Po wywołaniu metody MQGET , która określa tę opcję, komunikat nie jest już uwzględniany przez żadne otwarte uchwyty w zestawie współpracujących uchwytów jako oznaczony dla zestawu współpracującego. Komunikat jest nadal traktowany jako oznaczony na poziomie uchwytu, jeśli został oznaczony na poziomie uchwytu przed tym wywołaniem.

Użycie parametru MQGMO_UNMARK_BROWSE_CO_OP jest poprawne tylko w przypadku, gdy uchwyt został zwrócony przez pomyślne wywołanie metody MQOPEN z opcją MQOO_CO_OP. Operacja MQGET powiedzie się, nawet jeśli komunikat nie jest uznawany za oznaczony przez współpracujący zestaw uchwytów.

Parametr MQGMO_UNMARK_BROWSE_CO_OP nie jest poprawny w przypadku wywołania MQGET bez przeglądania lub z jedną z następujących opcji:

- 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
Po wywołaniu metody MQGET , która określa tę opcję, znaleziony komunikat nie jest już uznawany za oznaczony przez ten uchwyt.

Wywołanie powiedzie się, nawet jeśli komunikat nie jest oznaczony dla tego uchwytu.

Ta opcja nie jest poprawna w przypadku wywołania funkcji MQGET bez przeglądania lub z jedną z następujących opcji:

- 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

Opcje blokowania

Następujące opcje dotyczą blokowania komunikatów w kolejce:

BLOKADA MQGMO_LOCK
Zablokuj przeglądany komunikat, aby stał się niewidoczny dla każdego innego uchwytu otwartego dla kolejki. Opcję można podać tylko wtedy, gdy podano jedną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Dla każdego uchwytu kolejki można zablokować tylko jeden komunikat. Komunikat może być komunikatem logicznym lub komunikatem fizycznym:

- Jeśli zostanie podana wartość MQGMO_COMPLETE_MSG, wszystkie segmenty komunikatu, które tworzą komunikat logiczny, zostaną zablokowane w uchwycie kolejki. Wszystkie komunikaty muszą być obecne w kolejce i dostępne do pobrania.
- Jeśli parametr MQGMO_COMPLETE_MSGzostanie pominięty, do uchwytu kolejki zostanie zablokowany tylko jeden komunikat fizyczny. Jeśli ten komunikat jest segmentem komunikatu logicznego, zablokowany segment uniemożliwia innym aplikacjom pobranie lub przeglądanie komunikatu logicznego za pomocą programu MQGMO_COMPLETE_MSG .

Zablokowanym komunikatem jest zawsze ten, który znajduje się pod kursorem przeglądania. Komunikat można usunąć z kolejki za pomocą późniejszego wywołania MQGET , które określa opcję MQGMO_MSG_UNDER_CURSOR . Inne wywołania MQGET używające uchwytu kolejki mogą również usunąć komunikat (na przykład wywołanie, które określa identyfikator zablokowanego komunikatu).

Jeśli wywołanie zwróci kod zakończenia MQCC_FAILEDlub MQCC_WARNING z kodem przyczyny MQRC_TRUNCATED_MSG_FAILED, komunikat nie zostanie zablokowany.

Jeśli aplikacja nie usunie komunikatu z kolejki, blokada zostanie zwolniona przez jedno z następujących działań:

- Wywołanie innego wywołania MQGET dla tego uchwytu, z określeniem MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT. Blokada jest zwalniana, jeśli wywołanie zostanie zakończone z wartością MQCC_OK lub MQCC_WARNING. Komunikat pozostaje zablokowany, jeśli wywołanie zostanie zakończone z wartością MQCC_FAILED. Istnieją jednak wyjątki:
  - Komunikat nie jest odblokowywany, jeśli funkcja MQCC_WARNING została zwrócona z wartością MQRC_TRUNCATED_MSG_FAILED.
  - Komunikat zostanie odblokowany, jeśli funkcja MQCC_FAILED zostanie zwrócona z wartością MQRC_NO_MSG_AVAILABLE.

  Jeśli zostanie podana wartość MQGMO_LOCK, zwracany komunikat jest zablokowany. Jeśli parametr MQGMO_LOCKzostanie pominięty, po wywołaniu nie będzie żadnego zablokowanego komunikatu.

  Jeśli zostanie podana wartość MQGMO_WAITi żaden komunikat nie będzie natychmiast dostępny, oryginalny komunikat zostanie odblokowany przed rozpoczęciem oczekiwania.
- Wydanie kolejnego wywołania MQGET dla tego uchwytu, z wartością MQGMO_BROWSE_MSG_UNDER_CURSOR, bez wartości MQGMO_LOCK. Blokada jest zwalniana, jeśli wywołanie zostanie zakończone z wartością MQCC_OK lub MQCC_WARNING. Komunikat pozostaje zablokowany, jeśli wywołanie zostanie zakończone z wartością MQCC_FAILED. Jednak zastosowanie ma następujący wyjątek:
  - Komunikat nie jest odblokowywany, jeśli funkcja MQCC_WARNING została zwrócona z wartością MQRC_TRUNCATED_MSG_FAILED.
- Wykonanie kolejnego wywołania MQGET dla tego uchwytu za pomocą komendy MQGMO_UNLOCK.

- Wywołanie metody MQCLOSE z użyciem uchwytu. Wartość MQCLOSE może być niejawna, co jest spowodowane zakończeniem działania aplikacji.

Nie jest wymagana żadna specjalna opcja MQOPEN , aby określić opcję MQGMO_LOCKinną niż MQOO_BROWSE, która jest wymagana do określenia towarzyszącej opcji przeglądania.

Wartość MQGMO_LOCK nie jest poprawna z żadną z następujących opcji:

- MQGMO_MARK_SKIP_BACKOUT
- MQGMO_SYNCPOINT
- MQGMO_SYNCPOINT_IF_PERSISTENT
- MQGMO_UNLOCK

MQGMO_UNLOCK (odblokowanie MQGMO)
Komunikat, który ma zostać odblokowany, musi być wcześniej zablokowany przez wywołanie MQGET z opcją MQGMO_LOCK . Jeśli dla tego uchwytu nie ma zablokowanego komunikatu, wywołanie kończy się na MQCC_WARNING i MQRC_NO_MSG_LOCKED.

Parametry **MsgDesc**, **BufferLength**, **Buffer**i **DataLength** nie są sprawdzane ani zmieniane, jeśli zostanie podany parametr MQGMO_UNLOCK. W pliku *Buffer*nie jest zwracany żaden komunikat.

Nie jest wymagana żadna specjalna opcja otwierania w celu określenia wartości MQGMO_UNLOCK (chociaż do wydania żądania blokady w pierwszej kolejności potrzebna jest wartość MQOO_BROWSE).

Ta opcja nie jest poprawna dla żadnych opcji z wyjątkiem następujących:

- MQGMO_NO_WAIT
- MQGMO_NO_SYNCPOINT

Obie te opcje są przyjmowane niezależnie od tego, czy zostały określone, czy nie.

Opcje danych komunikatu

Następujące opcje odnoszą się do przetwarzania danych komunikatu, gdy komunikat jest odczytywany z kolejki:

MQGMO_ACCEPT_TRUNCATED_MSG
Jeśli bufor komunikatów jest zbyt mały, aby pomieścić cały komunikat, należy zezwolić na zapełnienie buforu przez wywołanie MQGET . Program MQGET zapełnia bufor jak najwięcej komunikatów, które może wysłać. Generuje kod zakończenia ostrzeżenia i kończy przetwarzanie. Oznacza to, że:

- Podczas przeglądania komunikatów kursor przeglądania jest przenoszony do zwróconego komunikatu.
- Podczas usuwania komunikatów zwracany komunikat jest usuwany z kolejki.
- Jeśli nie wystąpi żaden inny błąd, zwracany jest kod przyczyny MQRC_TRUNCATED_MSG_ACCEPTED.

Bez tej opcji bufor jest nadal wypełniony jak najwięcej komunikatów, które może pomieścić. Został wygenerowany kod zakończenia ostrzeżenia, ale przetwarzanie nie zostało zakończone. Oznacza to, że:

- Podczas przeglądania komunikatów kursor przeglądania nie jest zaawansowany.
- Podczas usuwania komunikatów komunikat nie jest usuwany z kolejki.
- Jeśli nie wystąpi inny błąd, zwracany jest kod przyczyny MQRC_TRUNCATED_MSG_FAILED .

MQGMO_CONVERT
Ta opcja przekształca dane aplikacji w komunikacie w taki sposób, aby były zgodne z wartościami CodedCharSetId i Encoding określonymi w parametrze **MsgDesc** w wywołaniu MQGET . Dane są przekształcane przed skopiowaniem do parametru **Buffer** .

Pole Format określone podczas umieszczania komunikatu jest przyjmowane przez proces konwersji w celu zidentyfikowania rodzaju danych w komunikacie. Dane komunikatu są przekształcane przez

menedżer kolejek dla formatów wbudowanych i przez program użytkownika dla innych formatów. Szczegółowe informacje na temat wyjścia konwersji danych zawiera sekcja "Wyjście konwersji danych" na stronie 937 .

- Jeśli konwersja powiedzie się, pola CodedCharSetId i Encoding określone w parametrze **MsgDesc** pozostaną niezmienione po powrocie z wywołania MQGET .
- Jeśli konwersja nie powiedzie się, zwracane są nieprzekształcone dane komunikatu. Pola CodedCharSetId i Encoding w pliku MsgDesc są ustawiane na wartości dla nieprzekształconego komunikatu. W tym przypadku kod zakończenia to MQCC_WARNING .

W obu przypadkach pola te opisują identyfikator zestawu znaków i kodowanie danych komunikatu, które są zwracane w parametrze **Buffer** .

Listę nazw formatów, dla których menedżer kolejek wykonuje konwersję, zawiera pole *Format* opisane w sekcji "MQMD-deskryptor komunikatu" na stronie 432 .

Opcje grupy i segmentu

Poniższe opcje odnoszą się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Przed opisami opcji znajdują się definicje ważnych terminów:

Komunikat fizyczny
Komunikat fizyczny to najmniejsza jednostka informacji, którą można umieścić w kolejce lub z niej usunąć. Często odpowiada to informacjom określonym lub pobranym w pojedynczym wywołaniu MQPUT, MQPUT1lub MQGET . Każdy komunikat fizyczny ma własny deskryptor komunikatu MQMD. Zazwyczaj komunikaty fizyczne są rozróżniane na podstawie różnych wartości identyfikatora komunikatu, pola MsgId w strukturze MQMD. Menedżer kolejek nie wymusza różnych wartości.

Komunikat logiczny
Komunikat logiczny jest pojedynczą jednostką informacji o aplikacji. W przypadku braku ograniczeń systemowych komunikat logiczny jest taki sam jak komunikat fizyczny. Jeśli komunikaty logiczne są duże, ograniczenia systemowe mogą spowodować, że wskazane lub konieczne będzie podzielenie komunikatu logicznego na dwa lub więcej komunikatów fizycznych, zwanych segmentami.

Komunikat logiczny, który został segmentowany, składa się z dwóch lub większej liczby komunikatów fizycznych, które mają ten sam identyfikator grupy o wartości innej niż NULL (pole GroupId w strukturze MQMD). Mają ten sam numer kolejny komunikatu, pole MsgSeqNumber w strukturze MQMD. Segmenty są rozróżniane na podstawie różnych wartości przesunięcia segmentu Offset w polu MQMD. Przesunięcie segmentu to przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dozwolona przez aplikację wysyłającą, ma również identyfikator grupy o wartości innej niż NULL. W takim przypadku istnieje tylko jeden komunikat fizyczny o tym identyfikatorze grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają pusty identyfikator grupy (MQGI_NONE), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy
Grupa komunikatów jest zestawem jednego lub większej liczby komunikatów logicznych, które mają ten sam identyfikator grupy o wartości innej niż NULL. Komunikaty logiczne w grupie są rozróżniane przez różne wartości numeru kolejnego komunikatu. Numer kolejny jest liczbą całkowitą z zakresu od 1 do n, gdzie n jest liczbą komunikatów logicznych w grupie. Jeśli jeden lub więcej komunikatów logicznych jest segmentowanych, w grupie znajduje się więcej niż n komunikatów fizycznych.

MQGMO_LOGICAL_ORDER (KOLEJNA_KOLEJNA_STRUKTURA)
Parametr MQGMO_LOGICAL_ORDER określa kolejność, w jakiej komunikaty są zwracane przez kolejne wywołania funkcji MQGET dla uchwytu kolejki. Opcja musi być określona przy każdym wywołaniu.

Jeśli parametr MQGMO_LOGICAL_ORDER jest określony dla kolejnych wywołań MQGET dla tego samego uchwytu kolejki, komunikaty w grupach są zwracane w kolejności numerów kolejnych komunikatów. Segmenty komunikatów logicznych są zwracane w kolejności określonej przez

przesunięcia segmentów. Ta kolejność może różnić się od kolejności, w jakiej te komunikaty i segmenty występują w kolejce.

Uwaga: Określenie wartości MQGMO_LOGICAL_ORDER nie ma negatywnego wpływu na komunikaty, które nie należą do grup i nie są segmentami. W rezultacie takie komunikaty są traktowane tak, jakby każdy należał do grupy komunikatów składającej się tylko z jednego komunikatu. Można bezpiecznie określić wartość MQGMO_LOGICAL_ORDER podczas pobierania komunikatów z kolejek, które zawierają mieszankę komunikatów w grupach, segmentach komunikatów i niesegmentowanych komunikatach, które nie są w grupach.

Aby zwrócić komunikaty w wymaganej kolejności, menedżer kolejek zachowuje informacje o grupie i segmencie między kolejnymi wywołaniami funkcji MQGET . Informacje o grupie i segmencie identyfikują bieżącą grupę komunikatów i bieżący komunikat logiczny dla uchwytu kolejki. Identyfikuje również bieżącą pozycję w grupie i komunikat logiczny oraz określa, czy komunikaty są pobierane w ramach jednostki pracy. Ponieważ menedżer kolejek zachowuje te informacje, aplikacja nie musi ustawiać informacji o grupie i segmencie przed każdym wywołaniem funkcji MQGET . Oznacza to, że aplikacja nie musi ustawiać pól GroupId, MsgSeqNumberi Offset w MQMD. Jednak w każdym wywołaniu aplikacja musi poprawnie ustawić opcję MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT .

Po otwarciu kolejki nie ma bieżącej grupy komunikatów ani bieżącego komunikatu logicznego. Grupa komunikatów staje się bieżącą grupą komunikatów, gdy komunikat z flagą MQMF_MSG_IN_GROUP jest zwracany przez wywołanie MQGET . Jeśli w kolejnych wywołaniach określono parametr MQGMO_LOGICAL_ORDER , grupa ta pozostaje bieżącą grupą do momentu zwrócenia komunikatu, który zawiera:

- MQMF_LAST_MSG_IN_GROUP bez MQMF_SEGMENT (oznacza to, że ostatni komunikat logiczny w grupie nie jest segmentowany) lub
- MQMF_LAST_MSG_IN_GROUP z wartością MQMF_LAST_SEGMENT (oznacza to, że zwrócony komunikat jest ostatnim segmentem ostatniego komunikatu logicznego w grupie).

Po zwróceniu takiego komunikatu grupa komunikatów jest przerywana, a po pomyślnym zakończeniu wywołania funkcji MQGET nie ma już bieżącej grupy. W podobny sposób komunikat logiczny staje się bieżącym komunikatem logicznym, gdy komunikat z flagą MQMF_SEGMENT jest zwracany przez wywołanie MQGET . Komunikat logiczny jest przerywany po zwróceniu komunikatu z flagą MQMF_LAST_SEGMENT .

Jeśli nie określono żadnych kryteriów wyboru, kolejne wywołania funkcji MQGET zwracają w poprawnej kolejności komunikaty dla pierwszej grupy komunikatów w kolejce. Następnie zwracają one komunikaty dla drugiej grupy komunikatów i tak dalej, aż nie będą dostępne żadne komunikaty. Można wybrać konkretne grupy komunikatów, podając jedną lub więcej spośród następujących opcji w polu MatchOptions :

- MQMO_MATCH_MSG_ID
- MQMO_MATCH_CORREL_ID
- MQMO_MATCH_GROUP_ID

Jednak te opcje obowiązują tylko wtedy, gdy nie ma bieżącej grupy komunikatów ani komunikatu logicznego. Więcej informacji na ten temat zawiera opis pola MatchOptions w sekcji “MQGMO-opcje pobierania komunikatów” na stronie 376 .

Tabela 495 na stronie 403 przedstawia wartości pól MsgId, CorrelId, GroupId, MsgSeqNumberi Offset , które są wyszukiwane przez menedżer kolejek podczas próby znalezienia komunikatu do zwrócenia w wywołaniu MQGET . Reguły dotyczą zarówno usuwania komunikatów z kolejki, jak i przeglądania komunikatów w kolejce. W tabeli oznacza Tak lub Nie:

LOG ORD
Wskazuje, czy opcja MQGMO_LOGICAL_ORDER jest określona w wywołaniu.

Cur grp
Wskazuje, czy przed wywołaniem istnieje bieżąca grupa komunikatów.

Cur log msg
Wskazuje, czy przed wywołaniem istnieje bieżący komunikat logiczny.

Inne kolumny

Pokaż wartości, których szuka menedżer kolejek. Poprzedni oznacza wartość zwróconą dla pola w poprzednim komunikacie dla uchwytu kolejki.

Tabela 495. Opcje MQGET dotyczące komunikatów w grupach i segmentach komunikatów logicznych

Określone opcje	**Status grupy i komunikatu dziennika przed wywołaniem**		**Wartości, których szuka menedżer kolejek**				
LOGORD	**Curgrp**	**Curlogmsg**	*MsgId*	*CorrelId*	*GroupId*	*MsgSeqNumber*	*Offset*
Tak	Nie	Nie	Kontrolowane przez *MatchOptions*	Kontrolowane przez *MatchOptions*	Kontrolowane przez *MatchOptions*	1	0
Tak	Nie	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + poprzednia długość segmentu
Tak	Tak	Nie	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + poprzednia długość segmentu
Nie	Albo	Albo	Kontrolowane przez *MatchOptions*	Kontrolowane przez *MatchOptions*	Kontrolowane przez *MatchOptions*	Kontrolowane przez *MatchOptions*	Kontrolowane przez *MatchOptions*

Jeśli w kolejce znajduje się wiele grup komunikatów, które mogą zostać zwrócone, grupy są zwracane w kolejności określonej przez pozycję w kolejce pierwszego segmentu pierwszego komunikatu logicznego w każdej grupie. Oznacza to, że komunikaty fizyczne, które mają numer kolejny komunikatu równy 1 i przesunięcie równe 0, określają kolejność, w jakiej zwracane są zakwalifikowane grupy.

Opcja MQGMO_LOGICAL_ORDER wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy logiczny komunikat lub segment w grupie jest pobierany w ramach jednostki pracy, wszystkie inne logiczne komunikaty i segmenty w grupie muszą zostać pobrane w ramach jednostki pracy, jeśli używany jest ten sam uchwyt kolejki. Nie muszą one jednak być pobierane w ramach tej samej jednostki pracy. Umożliwia to podzielenie grupy komunikatów składającej się z wielu komunikatów fizycznych na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.
- Jeśli pierwszy logiczny komunikat lub segment w grupie nie jest pobierany w ramach jednostki pracy i używany jest ten sam uchwyt kolejki, żaden inny logiczny komunikat lub segment w grupie nie może zostać pobrany w ramach jednostki pracy.

Jeśli te warunki nie są spełnione, wywołanie MQGET kończy się niepowodzeniem z kodem przyczyny MQRC_INCONSISTENT_UOW.

Jeśli określono parametr MQGMO_LOGICAL_ORDER , wartość MQGMO podana w wywołaniu MQGET nie może być mniejsza niż MQGMO_VERSION_2, a wartość MQMD nie może być mniejsza niż

MQMD_VERSION_2. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_WRONG_GMO_VERSION lub MQRC_WRONG_MD_VERSION(w zależności od przypadku).

Jeśli parametr MQGMO_LOGICAL_ORDER nie jest określony dla kolejnych wywołań MQGET dla uchwytu kolejki, komunikaty są zwracane bez względu na to, czy należą do grup komunikatów, czy też są segmentami komunikatów logicznych. Oznacza to, że komunikaty lub segmenty z konkretnej grupy lub komunikatu logicznego mogą być zwracane w nieodpowiedniej kolejności lub mieszane z komunikatami lub segmentami z innych grup lub komunikatów logicznych albo z komunikatami, które nie są w grupach i nie są segmentami. W takiej sytuacji konkretne komunikaty zwracane przez kolejne wywołania funkcji MQGET są kontrolowane przez opcje MQMO_* określone w tych wywołaniach (szczegółowe informacje na temat tych opcji zawiera opis pola *MatchOptions* w sekcji "MQGMO-opcje pobierania komunikatów" na stronie 376).

Jest to technika, która może być używana do restartowania grupy komunikatów lub komunikatu logicznego w środku, po wystąpieniu awarii systemu. Po restarcie systemu aplikacja może ustawić odpowiednie wartości w polach GroupId, MsgSeqNumber, Offseti MatchOptions , a następnie wywołać funkcję MQGET z ustawioną wartością MQGMO_SYNCPOINT lub MQGMO_NO_SYNCPOINT , ale bez określania wartości MQGMO_LOGICAL_ORDER. Jeśli wywołanie powiedzie się, menedżer kolejek zachowa informacje o grupie i segmencie, a kolejne wywołania programu MQGET używające tego uchwytu kolejki będą mogły normalnie określać parametr MQGMO_LOGICAL_ORDER .

Informacje o grupie i segmencie, które menedżer kolejek przechowuje dla wywołania MQGET , są oddzielone od informacji o grupie i segmencie, które przechowuje dla wywołania MQPUT. Ponadto menedżer kolejek zachowuje osobne informacje dla:

- Wywołania MQGET , które usuwają komunikaty z kolejki.
- Wywołania funkcji MQGET , które przeglądają komunikaty w kolejce.

Dla każdego uchwytu kolejki aplikacja może łączyć wywołania MQGET , które określają MQGMO_LOGICAL_ORDER z wywołaniami MQGET , które tego nie robią. Należy jednak zwrócić uwagę na następujące kwestie:

- Jeśli parametr MQGMO_LOGICAL_ORDERzostanie pominięty, każde pomyślne wywołanie funkcji MQGET spowoduje, że menedżer kolejek ustawi informacje o zapisanej grupie i segmencie na wartości odpowiadające zwróconemu komunikatowi. To spowoduje zastąpienie istniejących informacji o grupie i segmencie zachowanych przez menedżer kolejek dla uchwytu kolejki. Modyfikowane są tylko informacje odpowiednie dla działania wywołania (przeglądanie lub usuwanie).
- Jeśli parametr MQGMO_LOGICAL_ORDERzostanie pominięty, wywołanie nie zakończy się niepowodzeniem, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może zakończyć się powodzeniem z kodem zakończenia MQCC_WARNING . Tabela 496 na stronie 405 przedstawia różne obserwacje, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia jest inny niż MQCC_OK, kod przyczyny może być jednym z następujących (w zależności od przypadku):
  - MQRC_INCOMPLETE_GROUP
  - MQRC_INCOMPLETE_MSG
  - MQRC_INCONSISTENT_UOW

  Uwaga: Menedżer kolejek nie sprawdza informacji o grupie i segmencie podczas przeglądania kolejki lub podczas zamykania kolejki, która została otwarta do przeglądania, ale nie została wprowadzona. W takich przypadkach kodem zakończenia jest zawsze MQCC_OK (przy założeniu braku innych błędów).

Tabela 496. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupie i segmencie

Bieżące połączenie to	Poprzednie połączenie było MQGET z MQGMO_LOGICAL_ORDER	Poprzednie połączenie było MQGET bez MQGMO_LOGICAL_ORDER
MQGET z MQGMO_LOGICAL_ORDER	MQCC_FAILED	MQCC_FAILED
MQGET bez MQGMO_LOGICAL_ORDER	MQCC_WARNING	MQCC_OK
MQCLOSE z niezakończoną grupą lub komunikatem logicznym	MQCC_WARNING	MQCC_OK

Aplikacje, które chcą pobierać komunikaty i segmenty w kolejności logicznej, powinny podać parametr MQGMO_LOGICAL_ORDER, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia z konieczności zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tymi informacjami. Jednak wyspecjalizowane aplikacje mogą wymagać większej kontroli niż ta, która jest udostępniana przez opcję MQGMO_LOGICAL_ORDER , i można to osiągnąć, nie podając tej opcji. Następnie aplikacja musi upewnić się, że pola MsgId, CorrelId, GroupId, MsgSeqNumberi Offset w MQMDoraz opcje MQMO_* w MatchOptions w MQGMOsą ustawione poprawnie przed każdym wywołaniem MQGET .

Na przykład aplikacja, która chce przekazywać odebrane komunikaty fizyczne, bez względu na to, czy są one w grupach, czy w segmentach komunikatów logicznych, nie może określać parametru MQGMO_LOGICAL_ORDER. W złożonej sieci z wieloma ścieżkami między wysyłającymi i odbierającymi menedżerami kolejek komunikaty fizyczne mogą pojawiać się w nieodpowiedniej kolejności. Nie określając ani parametru MQGMO_LOGICAL_ORDER, ani odpowiedniego parametru MQPMO_LOGICAL_ORDER w wywołaniu MQPUT, aplikacja przekazująca może pobrać i przekazać każdy komunikat fizyczny natychmiast po jego nadejściu, bez konieczności oczekiwania na następny komunikat w porządku logicznym.

Parametr MQGMO_LOGICAL_ORDER można określić z innymi opcjami MQGMO_* oraz z różnymi opcjami MQMO_* w odpowiednich okolicznościach (patrz poprzednia sekcja).

- z/OS W systemie z/OSta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć typ indeksu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, na który odwzorowana jest kolejka, musi być na poziomie CFLEVEL (3) lub wyższym.
- Ta opcja jest obsługiwana dla wszystkich kolejek lokalnych na następujących platformach:
  - AIX AIX
  - Linux Linux
  - IBM i IBM i
  - Windows Windows

  i dla systemu IBM MQ MQI clients połączonego z tymi systemami.

MQGMO_COMPLETE_MSG
Wywołanie MQGET może zwrócić tylko pełny komunikat logiczny. Jeśli komunikat logiczny jest segmentowany, menedżer kolejek składa segmenty i zwraca do aplikacji pełny komunikat logiczny. Fakt, że komunikat logiczny został segmentowany, nie jest widoczny dla aplikacji, która go pobiera.

Uwaga: Jest to jedyna opcja, która powoduje, że menedżer kolejek składa segmenty komunikatów. Jeśli ta opcja nie zostanie podana, segmenty będą zwracane indywidualnie do aplikacji, jeśli są obecne w kolejce (i spełniają inne kryteria wyboru określone w wywołaniu MQGET). Aplikacje, które nie chcą otrzymywać pojedynczych segmentów, muszą zawsze mieć podany parametr MQGMO_COMPLETE_MSG.

Aby użyć tej opcji, aplikacja musi udostępnić bufor, który jest wystarczająco duży, aby pomieścić cały komunikat, lub podać opcję MQGMO_ACCEPT_TRUNCATED_MSG .

Jeśli kolejka zawiera segmentowane komunikaty z brakującymi segmentami (na przykład dlatego, że zostały opóźnione w sieci i jeszcze nie dotarły), podanie parametru MQGMO_COMPLETE_MSG uniemożliwia pobranie segmentów należących do niekompletnych komunikatów logicznych. Jednak te segmenty komunikatów nadal mają wpływ na wartość atrybutu kolejki **CurrentQDepth** . Oznacza to, że mogą nie istnieć żadne możliwe do pobrania komunikaty logiczne, nawet jeśli wartość *CurrentQDepth* jest większa od zera.

W przypadku trwałych komunikatów menedżer kolejek może złożyć segmenty tylko w ramach jednostki pracy:

- Jeśli wywołanie funkcji MQGET działa w obrębie jednostki pracy zdefiniowanej przez użytkownika, używana jest ta jednostka pracy. Jeśli wywołanie nie powiedzie się podczas procesu ponownego składania, menedżer kolejek przywraca do kolejki wszystkie segmenty, które zostały usunięte podczas ponownego składania. Jednak niepowodzenie nie uniemożliwia pomyślnego zatwierdzenia jednostki pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika i nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy w czasie trwania wywołania. Jeśli wywołanie powiedzie się, menedżer kolejek automatycznie zatwierdza jednostkę pracy (aplikacja nie musi tego robić). Jeśli wywołanie nie powiedzie się, menedżer kolejek wycofa jednostkę pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika, ale istnieje jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek nie może złożyć ponownie. Jeśli komunikat nie wymaga ponownego składania, wywołanie nadal może zakończyć się powodzeniem. Jeśli jednak komunikat wymaga ponownego złożenia, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_UOW_NOT_AVAILABLE.

W przypadku nietrwałych komunikatów menedżer kolejek nie wymaga, aby jednostka pracy była dostępna na potrzeby ponownego składania.

Każdy komunikat fizyczny, który jest segmentem, ma własny deskryptor komunikatu. W przypadku segmentów tworzących pojedynczy komunikat logiczny większość pól w deskryptorze komunikatu jest taka sama dla wszystkich segmentów w komunikacie logicznym. Zwykle różnią się one tylko polami MsgId, Offseti MsgFlags w komunikacie logicznym. Jeśli jednak segment zostanie umieszczony w kolejce niedostarczonych komunikatów w pośrednim menedżerze kolejek, program obsługi DLQ pobiera komunikat, określając opcję MQGMO_CONVERT , co może spowodować zmianę zestawu znaków lub kodowania segmentu. Jeśli program obsługi DLQ pomyślnie wyśle segment w swoją stronę, segment może mieć zestaw znaków lub kodowanie różniące się od innych segmentów w komunikacie logicznym, gdy segment dotrze do docelowego menedżera kolejek.

Menedżer kolejek nie może złożyć komunikatu logicznego składającego się z segmentów, w których pola CodedCharSetId i Encoding różnią się od siebie, w jeden komunikat logiczny. Zamiast tego menedżer kolejek składa i zwraca kilka pierwszych kolejnych segmentów na początku komunikatu logicznego, które mają takie same identyfikatory zestawu znaków i kodowania, a wywołanie MQGET kończy się kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_INCONSISTENT_CCSIDS lub MQRC_INCONSISTENT_ENCODINGS, w zależności od przypadku. Dzieje się tak niezależnie od tego, czy określono parametr MQGMO_CONVERT . Aby pobrać pozostałe segmenty, aplikacja musi ponownie wywołać funkcję MQGET bez opcji MQGMO_COMPLETE_MSG , pobierając segmenty jeden po drugim. Parametr MQGMO_LOGICAL_ORDER może być używany do pobierania pozostałych segmentów w kolejności.

Aplikacja, która umieszcza segmenty, może również ustawić inne pola w deskryptorze komunikatu na wartości różniące się między segmentami. Nie ma jednak żadnej korzyści, jeśli aplikacja odbierająca pobiera komunikat logiczny za pomocą funkcji MQGMO_COMPLETE_MSG . Gdy menedżer kolejek ponownie składa komunikat logiczny, zwraca w deskryptorze komunikatu wartości z deskryptora komunikatu dla pierwszego segmentu. Jedynym wyjątkiem jest pole MsgFlags , które menedżer kolejek ustawia w celu wskazania, że złożony komunikat jest jedynym segmentem.

Jeśli dla komunikatu raportu określono wartość MQGMO_COMPLETE_MSG , menedżer kolejek wykonuje przetwarzanie specjalne. Menedżer kolejek sprawdza kolejkę, aby sprawdzić, czy wszystkie komunikaty raportu tego typu dotyczące różnych segmentów w komunikacie logicznym znajdują się w kolejce. Jeśli tak, można je pobrać jako pojedynczy komunikat, podając parametr MQGMO_COMPLETE_MSG. Aby było to możliwe, komunikaty raportu muszą zostać wygenerowane przez menedżer kolejek lub agent MCA obsługujący segmentację lub aplikację inicjującą musi zażądać co najmniej 100 bajtów danych komunikatu (należy określić odpowiednie opcje MQRO_*_WITH_DATA lub MQRO_*_WITH_FULL_DATA). Jeśli dla segmentu występuje mniej niż pełna ilość danych aplikacji, brakujące bajty są zastępowane przez wartości null w zwróconym komunikacie raportu.

Jeśli dla parametru MQGMO_COMPLETE_MSG określono wartość MQGMO_MSG_UNDER_CURSOR lub MQGMO_BROWSE_MSG_UNDER_CURSOR, kursor przeglądania musi być ustawiony na komunikacie, którego pole *Offset* w MQMD ma wartość 0. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_INVALID_MSG_UNDER_CURSOR.

MQGMO_COMPLETE_MSG oznacza MQGMO_ALL_SEGMENTS_AVAILABLE, która nie musi być określona.

Parametr MQGMO_COMPLETE_MSG można podać razem z dowolną inną opcją MQGMO_* oprócz opcji MQGMO_SYNCPOINT_IF_PERSISTENToraz z dowolną z opcji MQMO_* poza opcją MQMO_MATCH_OFFSET.

- z/OS W systemie z/OSta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć indeks typu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, do którego mapa kolejek musi być na poziomie CFLEVEL (3) lub wyższym.
- Na następujących platformach:
  - AIX AIX
  - IBM i IBM i
  - Linux Linux
  - Windows Windows

  i dla IBM MQ MQI clients połączonych z tymi systemami, ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

MQGMO_ALL_MSGS_AVAILABLE
Komunikaty w grupie stają się dostępne do pobrania tylko wtedy, gdy wszystkie komunikaty w grupie są dostępne. Jeśli kolejka zawiera grupy komunikatów z brakującymi niektórymi komunikatami (na przykład dlatego, że zostały one opóźnione w sieci i nie dotarły jeszcze do niej), podanie parametru MQGMO_ALL_MSGS_AVAILABLE uniemożliwia pobranie komunikatów należących do niekompletnych grup. Jednak te komunikaty nadal mają wpływ na wartość atrybutu kolejki **CurrentQDepth** . Oznacza to, że grupy komunikatów, które można pobrać, mogą nie istnieć, nawet jeśli wartość CurrentQDepth jest większa od zera. Jeśli nie ma innych komunikatów, które można pobrać, kod przyczyny MQRC_NO_MSG_AVAILABLE jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie parametru MQGMO_ALL_MSGS_AVAILABLE zależy od tego, czy określono również parametr MQGMO_LOGICAL_ORDER :

- Jeśli zostaną podane obie opcje, opcja MQGMO_ALL_MSGS_AVAILABLE ma wpływ tylko wtedy, gdy nie ma bieżącej grupy ani komunikatu logicznego. Jeśli istnieje bieżąca grupa lub komunikat logiczny, parametr MQGMO_ALL_MSGS_AVAILABLE jest ignorowany. Oznacza to, że produkt MQGMO_ALL_MSGS_AVAILABLE może pozostać włączony podczas przetwarzania komunikatów w porządku logicznym.
- Jeśli parametr MQGMO_ALL_MSGS_AVAILABLE zostanie określony bez parametru MQGMO_LOGICAL_ORDER, parametr MQGMO_ALL_MSGS_AVAILABLE zawsze będzie miał zastosowanie. Oznacza to, że opcja musi zostać wyłączona po usunięciu pierwszego komunikatu z grupy z kolejki, aby możliwe było usunięcie pozostałych komunikatów z grupy.

Pomyślne zakończenie wywołania MQGET z określeniem MQGMO_ALL_MSGS_AVAILABLE oznacza, że w momencie wywołania funkcji MQGET wszystkie komunikaty w grupie znajdowały się w kolejce. Należy jednak pamiętać, że inne aplikacje nadal mogą usuwać komunikaty z grupy (grupa nie jest zablokowana dla aplikacji, która pobiera pierwszy komunikat z grupy).

Jeśli ta opcja zostanie pominięta, komunikaty należące do grup mogą zostać odtworzone nawet wtedy, gdy grupa jest niekompletna.

MQGMO_ALL_MSGS_AVAILABLE oznacza MQGMO_ALL_SEGMENTS_AVAILABLE, która nie musi być określona.

Parametr MQGMO_ALL_MSGS_AVAILABLE można podać z innymi opcjami MQGMO_* oraz z dowolną z opcji MQMO_* .

- z/OS W systemie z/OSta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć indeks typu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, do którego mapa kolejek musi być na poziomie CFLEVEL (3) lub wyższym.
- Na następujących platformach:
  - AIX AIX
  - IBM i IBM i
  - Linux Linux
  - Windows Windows

  i dla IBM MQ MQI clients połączonych z tymi systemami, ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

MQGMO_ALL_SEGMENTS_AVAILABLE
Segmenty w komunikacie logicznym stają się dostępne do pobrania tylko wtedy, gdy wszystkie segmenty w komunikacie logicznym są dostępne. Jeśli kolejka zawiera segmentowane komunikaty z brakującymi niektórymi segmentami (na przykład dlatego, że zostały opóźnione w sieci i jeszcze nie dotarły), podanie parametru MQGMO_ALL_SEGMENTS_AVAILABLE uniemożliwia pobranie segmentów należących do niekompletnych komunikatów logicznych. Jednak segmenty te nadal mają wpływ na wartość atrybutu kolejki **CurrentQDepth** . Oznacza to, że mogą nie istnieć żadne możliwe do pobrania komunikaty logiczne, nawet jeśli wartość CurrentQDepth jest większa od zera. Jeśli nie ma innych komunikatów, które można pobrać, kod przyczyny MQRC_NO_MSG_AVAILABLE jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie parametru MQGMO_ALL_SEGMENTS_AVAILABLE zależy od tego, czy określono również parametr MQGMO_LOGICAL_ORDER :

- Jeśli zostaną podane obie opcje, opcja MQGMO_ALL_SEGMENTS_AVAILABLE ma wpływ tylko wtedy, gdy nie ma bieżącego komunikatu logicznego. Jeśli istnieje bieżący komunikat logiczny, parametr MQGMO_ALL_SEGMENTS_AVAILABLE jest ignorowany. Oznacza to, że produkt MQGMO_ALL_SEGMENTS_AVAILABLE może pozostać włączony podczas przetwarzania komunikatów w porządku logicznym.
- Jeśli parametr MQGMO_ALL_SEGMENTS_AVAILABLE zostanie określony bez parametru MQGMO_LOGICAL_ORDER, parametr MQGMO_ALL_SEGMENTS_AVAILABLE zawsze będzie miał zastosowanie. Oznacza to, że opcja musi zostać wyłączona po usunięciu z kolejki pierwszego segmentu komunikatu logicznego, aby można było usunąć pozostałe segmenty komunikatu logicznego.

Jeśli ta opcja nie zostanie podana, segmenty komunikatów mogą zostać pobrane nawet wtedy, gdy komunikat logiczny jest niekompletny.

Oba te elementy (MQGMO_COMPLETE_MSG i MQGMO_ALL_SEGMENTS_AVAILABLE) wymagają, aby wszystkie segmenty były dostępne, zanim będzie można je pobrać, ale pierwszy z nich zwraca pełny komunikat, podczas gdy drugi z nich umożliwia pobieranie segmentów pojedynczo.

Jeśli dla komunikatu raportu określono parametr MQGMO_ALL_SEGMENTS_AVAILABLE , menedżer kolejek sprawdza kolejkę, aby sprawdzić, czy istnieje co najmniej jeden komunikat raportu dla każdego z segmentów, które tworzą pełny komunikat logiczny. Jeśli istnieje, warunek MQGMO_ALL_SEGMENTS_AVAILABLE jest spełniony. Jednak menedżer kolejek nie sprawdza *typu* komunikatów raportu, dlatego w komunikatach raportu dotyczących segmentów komunikatu logicznego może występować mieszanka typów raportów. Z tego powodu powodzenie komendy MQGMO_ALL_SEGMENTS_AVAILABLE nie oznacza, że komenda MQGMO_COMPLETE_MSG zakończy się pomyślnie. Jeśli istnieje kombinacja typów raportów dla segmentów konkretnego komunikatu logicznego, te komunikaty muszą być pobierane pojedynczo.

Parametr MQGMO_ALL_SEGMENTS_AVAILABLE można podać z dowolną inną opcją MQGMO_* i dowolną z opcji MQMO_* .

- W systemie z/OSta opcja jest obsługiwana dla kolejek prywatnych i współużytkowanych, ale kolejka musi mieć indeks typu MQIT_GROUP_ID. W przypadku kolejek współużytkowanych obiekt CFSTRUCT, do którego mapa kolejek musi być na poziomie CFLEVEL (3) lub wyższym.
- Na następujących platformach:
  - AIX
  - IBM i
  - Linux
  - Windows

  i dla IBM MQ MQI clients połączonych z tymi systemami, ta opcja jest obsługiwana dla wszystkich kolejek lokalnych.

Opcje właściwości

Następujące opcje odnoszą się do właściwości komunikatu:

MQGMO_PROPERTIES_AS_Q_DEF

Właściwości komunikatu, z wyjątkiem tych, które są zawarte w deskryptorze komunikatu (lub rozszerzeniu), powinny być reprezentowane jako zdefiniowane przez atrybut kolejki **PropertyControl** . Jeśli podano parametr MsgHandle , ta opcja jest ignorowana, a właściwości komunikatu są dostępne za pośrednictwem pliku MsgHandle, chyba że atrybut kolejki **PropertyControl** ma wartość MQPROP_FORCE_MQRFH2.

Jest to działanie domyślne w sytuacji, gdy nie są określone opcje właściwości.

MQGMO_PROPERTIES_IN_HANDLE

Właściwości komunikatu powinny być dostępne za pośrednictwem MsgHandle. Jeśli nie udostępniono uchwytu komunikatu, wywołanie zakończy się niepowodzeniem z następującej przyczyny: MQRC_HMSG_ERROR.

Uwaga: Jeśli komunikat zostanie później odczytany przez aplikację, która nie tworzy uchwytu komunikatu, menedżer kolejek umieści wszystkie właściwości komunikatu w strukturze MQRFH2 . Może się okazać, że obecność nieoczekiwanego nagłówka MQRFH2 zakłóca działanie istniejącej aplikacji.

MQGMO_NO_PROPERTIES

Nie zostaną pobrane żadne właściwości komunikatu, z wyjątkiem tych, które są zawarte w deskryptorze komunikatu (lub rozszerzeniu). Jeśli zostanie podany parametr MsgHandle , zostanie on zignorowany.

MQGMO_PROPERTIES_FORCE_MQRFH2

Właściwości komunikatu, z wyjątkiem tych, które są zawarte w deskryptorze komunikatu (lub rozszerzeniu), powinny być reprezentowane przy użyciu nagłówków MQRFH2 . Zapewnia to kompatybilność z wcześniejszymi wersjami aplikacji, które oczekują na pobranie właściwości, ale

nie mogą zostać zmienione w celu użycia uchwytów komunikatów. Jeśli zostanie podany parametr MsgHandle , zostanie on zignorowany.

MQGMO_PROPERTIES_COMPATIBILITY
Jeśli komunikat zawiera właściwość z przedrostkiem **"mcd."**, **"jms."**, **"usr."**lub **"mqext."**, wszystkie właściwości komunikatu są dostarczane do aplikacji w nagłówku MQRFH2 . W przeciwnym razie wszystkie właściwości komunikatu z wyjątkiem tych, które są zawarte w deskryptorze komunikatu lub w rozszerzeniu, są usuwane i nie są już dostępne dla aplikacji.

Opcja domyślna

Jeśli żadna z opisanych opcji nie jest wymagana, można użyć następującej opcji:

MQGMO_NONE
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. MQGMO_NONE pomaga w dokumentacji programu; nie jest planowane użycie tej opcji z żadną inną, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Wartością początkową pola Options jest MQGMO_NO_WAIT plus MQGMO_PROPERTIES_AS_Q_DEF.

WaitInterval (MQLONG) dla MQGMO

Jest to przybliżony czas w milisekundach, przez który wywołanie MQGET oczekuje na nadejście odpowiedniego komunikatu (czyli komunikatu spełniającego kryteria wyboru określone w parametrze **MsgDesc** wywołania MQGET).

Ważne: Nie ma oczekiwania ani opóźnienia, jeśli odpowiedni komunikat jest dostępny natychmiast.

Więcej szczegółów zawiera opis pola *MsgId* w sekcji “MQMD-deskryptor komunikatu” na stronie 432). Jeśli po upłynięciu tego czasu nie został odebrany odpowiedni komunikat, wywołanie zostanie zakończone komunikatem MQCC_FAILED i kodem przyczyny MQRC_NO_MSG_AVAILABLE.

W systemie z/OSna czas oczekiwania wywołania MQGET mają wpływ kwestie związane z ładowaniem systemu i planowaniem pracy. Może on być różny w zależności od wartości określonej dla parametru *WaitInterval* i około 100 milisekund większy niż wartość parametru *WaitInterval*.

Opcja *WaitInterval* jest używana w połączeniu z opcją MQGMO_WAIT lub MQGMO_SET_SIGNAL. Jeśli żadna z tych opcji nie zostanie podana, zostanie ona zignorowana. Jeśli zostanie podana jedna z tych wartości, wartość *WaitInterval* musi być większa lub równa zero lub następująca wartość specjalna:

MQWI_UNLIMITED
Nieograniczony czas oczekiwania.

Wartością początkową tego pola jest 0.

Signal1 (MQLONG) dla MQGMO

Jest to pole wejściowe używane tylko w połączeniu z opcją MQGMO_SET_SIGNAL. Identyfikuje ono sygnał, który ma zostać dostarczony, gdy komunikat jest dostępny.

Uwaga: Typ danych i użycie tego pola są określane przez środowisko. Z tego powodu aplikacje, które mają być przesyłane między różnymi środowiskami, nie mogą używać sygnałów.

- W systemie z/OSpole to musi zawierać adres bloku kontroli zdarzeń (Event Control Block-EBC). EBC musi zostać rozliczony przez aplikację przed wywołaniem MQGET. Pamięć zawierająca EBC nie może być zwolniona do czasu zamknięcia kolejki. EBC jest księgowany przez menedżera kolejek z jednym z opisanych kodów zakończenia sygnału. Te kody zakończenia są ustawiane w bitach od 2 do 31 EBC, obszar zdefiniowany w makrze odwzorowania z/OS IHAECB jako kod zakończenia użytkownika.
- We wszystkich innych środowiskach jest to pole zastrzeżone; jego wartość nie jest istotna.

Kody zakończenia sygnału są następujące:

MQEC_MSG_ODEBRANY
W kolejce został odebrany odpowiedni komunikat. Ten komunikat nie został zarezerwowany dla programu wywołującego. Należy wysłać drugie żądanie MQGET, ale inna aplikacja może pobrać komunikat przed wykonaniem drugiego żądania.

MQEC_WAIT_INTERVAL_EXPIRED:
Podany parametr *WaitInterval* utracił ważność bez odpowiedniego komunikatu.

MQEC_WAIT_ANULOWANA
Oczekiwanie zostało anulowane z nieokreślonej przyczyny (takiej jak zakończenie menedżera kolejek lub wyłączenie kolejki). Ponownie wyślij żądanie, jeśli chcesz uzyskać dalszą diagnozę.

MQEC_Q_MGR_QUIESCING,
Oczekiwanie zostało anulowane, ponieważ menedżer kolejek wszedł w stan wyciszania (w wywołaniu MQGET określono opcję MQGMO_FAIL_IF_QUIESCING).

MQEC_CONNECTION_QUIESCING,
Oczekiwanie zostało anulowane, ponieważ połączenie weszło w stan wyciszania (w wywołaniu MQGET określono opcję MQGMO_FAIL_IF_QUIESCING).

Początkowa wartość tego pola jest określana przez środowisko:

- W systemie z/OSwartością początkową jest wskaźnik pusty.
- We wszystkich innych środowiskach wartością początkową jest 0.

Signal2 (MQLONG) dla MQGMO

Jest to pole wejściowe używane tylko w połączeniu z opcją MQGMO_SET_SIGNAL. Jest to pole zastrzeżone; jego wartość nie jest istotna.

Wartością początkową tego pola jest 0.

ResolvedQName (MQCHAR48) dla obiektu MQGMO

Jest to pole wyjściowe, które menedżer kolejek ustawia na nazwę lokalną kolejki, z której został pobrany komunikat, zgodnie z definicją w menedżerze kolejek lokalnych. Różni się to od nazwy użytej do otwarcia kolejki, jeśli:

- Kolejka aliasowa została otwarta (w takim przypadku zwracana jest nazwa kolejki lokalnej, do której alias został przetłumaczony), lub
- Otwarto kolejkę modelową (w tym przypadku zwracana jest nazwa dynamicznej kolejki lokalnej).

Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

MatchOptions (MQLONG) dla MQGMO

Te opcje umożliwiają aplikacji wybór pól w parametrze **MsgDesc** , które mają być używane do wybierania komunikatu zwracanego przez wywołanie MQGET. Aplikacja ustawia wymagane opcje w tym polu, a następnie ustawia odpowiednie pola w parametrze **MsgDesc** na wartości wymagane dla tych pól. Tylko komunikaty, które mają te wartości w deskryptorze MQMD dla komunikatu, są kandydatami do pobrania przy użyciu tego parametru **MsgDesc** w wywołaniu MQGET. Pola, dla których nie określono odpowiedniej opcji dopasowania, są ignorowane podczas wybierania komunikatu, który ma zostać zwrócony. Jeśli w wywołaniu MQGET nie określono żadnych kryteriów wyboru (komunikat *any* jest akceptowalny), należy ustawić parametr *MatchOptions* na wartość MQMO_NONE.

- W systemie z/OSkryteria wyboru, które mogą być używane, mogą być ograniczone przez typ indeksu używanego dla kolejki. Więcej informacji na ten temat zawiera opis atrybutu kolejki **IndexType** .

Jeśli określono parametr MQGMO_LOGICAL_ORDER, tylko niektóre komunikaty mogą zostać zwrócone przez następne wywołanie MQGET:

- Jeśli nie ma bieżącej grupy lub komunikatu logicznego, tylko komunikaty, które mają *MsgSeqNumber* równe 1 i *Offset* równe 0 , są zakwalifikowane do zwrotu. W takiej sytuacji można użyć jednej lub kilku

z następujących opcji dopasowania, aby wybrać, który z zakwalifikowanych komunikatów ma zostać zwrócony:

  - MQMO_MATCH_MSG_ID
  - MQMO_MATCH_CORREL_ID
  - MQMO_MATCH_GROUP_ID (identyfikator grupy zgodności MQ)
- Jeśli istnieje bieżąca grupa lub komunikat logiczny, tylko następny komunikat w grupie lub następnym segmencie w komunikacie logicznym jest zakwalifikowany do zwrotu i nie można go zmienić przez podanie opcji MQMO_ *.

W obu powyższych przypadkach można określić opcje dopasowania, które nie mają zastosowania, ale wartość odpowiedniego pola w parametrze **MsgDesc** musi być zgodna z wartością odpowiedniego pola w komunikacie, który ma zostać zwrócony. Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MATCH_OPTIONS_ERROR, jeśli ten warunek nie został spełniony.

Parametr *MatchOptions* jest ignorowany, jeśli określono wartość MQGMO_MSG_UNDER_CURSOR lub MQGMO_BROWSE_MSG_UNDER_CURSOR.

Pobieranie komunikatów na podstawie właściwości komunikatu nie jest wykonywane przy użyciu opcji dopasowania. Więcej informacji na ten temat zawiera sekcja "SelectionString (MQCHARV) dla usługi MQOD" na stronie 507.

Można określić jedną lub więcej spośród następujących opcji dopasowania:

MQMO_MATCH_MSG_ID
Komunikat do pobrania musi mieć identyfikator komunikatu zgodny z wartością pola *MsgId* w parametrze **MsgDesc** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja zostanie pominięta, pole *MsgId* w parametrze **MsgDesc** zostanie zignorowane, a każdy identyfikator komunikatu będzie zgodny.

Uwaga: Identyfikator komunikatu MQMI_NONE jest wartością specjalną, która jest zgodna z dowolnym identyfikatorem komunikatu w deskryptorze MQMD. Oznacza to, że określenie MQMO_MATCH_MSG_ID z MQMI_NONE jest takie samo, jak niepodanie MQMO_MATCH_MSG_ID.

MQMO_MATCH_CORREL_ID
Komunikat do pobrania musi mieć identyfikator korelacji zgodny z wartością pola *CorrelId* w parametrze **MsgDesc** wywołania MQGET. To dopasowanie jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator komunikatu).

Jeśli ta opcja zostanie pominięta, pole *CorrelId* w parametrze **MsgDesc** zostanie zignorowane, a każdy identyfikator korelacji będzie zgodny.

Uwaga: Identyfikator korelacji MQCI_NONE jest wartością specjalną, która jest zgodna z *dowolnym* identyfikatorem korelacji w deskryptorze MQMD dla komunikatu. Oznacza to, że określenie MQMO_MATCH_CORREL_ID z MQCI_NONE jest takie samo, jak niepodanie MQMO_MATCH_CORREL_ID.

MQMO_MATCH_GROUP_ID (identyfikator grupy zgodności MQ)
Komunikat do pobrania musi mieć identyfikator grupy zgodny z wartością pola *GroupId* w parametrze **MsgDesc** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja zostanie pominięta, pole *GroupId* w parametrze **MsgDesc** zostanie zignorowane, a każdy identyfikator grupy będzie zgodny.

Uwaga: Identyfikator grupy MQGI_NONE jest wartością specjalną, która jest zgodna z *dowolnym* identyfikatorem grupy w deskryptorze MQMD dla komunikatu. Dlatego podanie wartości MQMO_MATCH_GROUP_ID w parametrze MQGI_NONE jest takie samo, jak niepodanie wartości MQMO_MATCH_GROUP_ID.

MQMO_MATCH_MSG_SEQ_NUMBER
Komunikat do pobrania musi mieć numer kolejny komunikatu zgodny z wartością pola *MsgSeqNumber* w parametrze **MsgDesc** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator grupy).

Jeśli ta opcja zostanie pominięta, pole *MsgSeqNumber* w parametrze **MsgDesc** zostanie zignorowane, a każdy numer kolejny komunikatu będzie zgodny.

MQMO_ZGODNE_PRZESUNIĘCIE
Komunikat do pobrania musi mieć przesunięcie zgodne z wartością pola *Offset* w parametrze **MsgDesc** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład numeru kolejnego komunikatu).

Jeśli ta opcja nie zostanie pominięta, pole *Offset* w parametrze **MsgDesc** zostanie zignorowane, a każde przesunięcie będzie zgodne.

- Ta opcja nie jest obsługiwana w systemie z/OS.

MQMO_MATCH_MSG_TOKEN,
Komunikat do pobrania musi mieć znacznik komunikatu zgodny z wartością pola *MsgToken* w strukturze MQGMO określonej w wywołaniu MQGET.

Opcję tę można określić dla wszystkich kolejek lokalnych. Jeśli zostanie on określony dla kolejki, która ma parametr *IndexType* o wartości MQIT_MSG_TOKEN (kolejka zarządzana przez WLM), nie można określić żadnych innych zgodnych opcji dla parametru MQMO_MATCH_MSG_TOKEN.

Nie można określić opcji MQMO_MATCH_MSG_TOKEN z opcją MQGMO_WAIT lub MQGMO_SET_SIGNAL. Jeśli aplikacja ma oczekiwać na przybycie komunikatu do kolejki, w której parametr *IndexType* ma wartość MQIT_MSG_TOKEN, należy określić parametr MQMO_NONE.

Jeśli ta opcja zostanie pominięta, pole *MsgToken* w MQGMO zostanie zignorowane, a każdy znacznik komunikatu będzie zgodny.

Jeśli nie zostanie podana żadna z opisanych opcji, można użyć następującej opcji:

MQMO_BRAK
Nie należy używać zgodnych elementów podczas wybierania komunikatu, który ma zostać zwrócony. Wszystkie komunikaty w kolejce są zakwalifikowane do pobrania (ale podlegają kontroli przez opcje MQGMO_ALL_MSGS_AVAILABLE, MQGMO_ALL_SEGMENTS_AVAILABLE i MQGMO_COMPLETE_MSG).

MQMO_NONE pomaga w dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją MQMO_ *, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Jest to pole wejściowe. Wartością początkową tego pola jest MQMO_MATCH_MSG_ID z MQMO_MATCH_CORREL_ID. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

Uwaga: Wartość początkowa pola *MatchOptions* jest definiowana w celu zachowania zgodności z wcześniejszymi menedżerami kolejek systemu MQSeries . Jednak podczas odczytywania serii komunikatów z kolejki bez użycia kryteriów wyboru ta wartość początkowa wymaga, aby aplikacja zresetować pola *MsgId* i *CorrelId* do wartości MQMI_NONE i MQCI_NONE przed każdym wywołaniem MQGET. Aby uniknąć konieczności resetowania parametrów *MsgId* i *CorrelId* , należy ustawić parametr *Version* na wartość MQGMO_VERSION_2, a parametr *MatchOptions* na wartość MQMO_NONE.

Pojęcia pokrewne
Selektory komunikatów w usłudze JMS

GroupStatus (MQCHAR) dla MQGMO
Ta opcja wskazuje, czy pobrany komunikat należy do grupy.

Ma jedną z następujących wartości:

GRUPA MQGS_NOT_IN_
Komunikat nie należy do grupy.

MQGS_MSG_IN_GROUP
Komunikat znajduje się w grupie, ale nie jest ostatnim w grupie.

MQGS_LAST_MSG_IN_GROUP
Komunikat jest ostatnim w grupie.

Jest to również wartość zwracana, jeśli grupa składa się tylko z jednego komunikatu.

Jest to pole wyjściowe. Wartością początkową tego pola jest MQGS_NOT_IN_GROUP. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

SegmentStatus (MQCHAR) dla MQGMO

Jest to flaga wskazująca, czy pobrany komunikat jest segmentem komunikatu logicznego. Ma jedną z następujących wartości:

MQSS_NOT_A_SEGMENT (MQSS_NOT_A)
Komunikat nie jest segmentem.

SEGMENT_MQSS
Komunikat jest segmentem, ale nie jest ostatnim segmentem komunikatu logicznego.

MQSS_OSTATNI_SEGMENT
Komunikat jest ostatnim segmentem komunikatu logicznego.

Jest to również wartość zwracana, jeśli komunikat logiczny składa się tylko z jednego segmentu.

W systemie z/OSmenedżer kolejek zawsze ustawia w tym polu wartość MQSS_NOT_A_SEGMENT.

Jest to pole wyjściowe. Wartością początkową tego pola jest MQSS_NOT_A_SEGMENT. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

Segmentacja (MQCHAR) dla MQGMO

Jest to flaga wskazująca, czy dla pobieranego komunikatu dozwolona jest dalsza segmentacja. Ma jedną z następujących wartości:

MQSEG_INHIBITED
Segmentacja niedozwolona.

DOZWOLONY_SEGMENT_MQB
Segmentacja jest dozwolona.

W systemie z/OSmenedżer kolejek zawsze ustawia w tym polu wartość MQSEG_INHIBITED.

Jest to pole wyjściowe. Wartością początkową tego pola jest MQSEG_INHIBITED. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

Reserved1 (MQCHAR) dla MQGMO

Jest to pole zastrzeżone. Wartość początkowa tego pola jest znakiem odstępu. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_2.

MsgToken (MQBYTE16) w produkcie MQGMO

Pole MsgToken -struktura MQGMO. To pole jest używane przez menedżer kolejek do jednoznacznego identyfikowania komunikatu.

Jest to łańcuch bajtów generowany przez menedżer kolejek w celu jednoznacznego zidentyfikowania komunikatu w kolejce. Znacznik komunikatu jest generowany, gdy komunikat jest umieszczany po raz pierwszy w menedżerze kolejek i pozostaje z komunikatem do czasu, aż komunikat zostanie trwale usunięty z menedżera kolejek, chyba że menedżer kolejek zostanie zrestartowany.

Gdy komunikat jest usuwany z kolejki, *MsgToken* , który zidentyfikował tę instancję komunikatu, nie jest już poprawny i nigdy nie jest ponownie wykorzystywany. Jeśli menedżer kolejek został zrestartowany, *MsgToken* , który zidentyfikował komunikat w kolejce przed restartem, może nie być poprawny po restarcie. Jednak parametr *MsgToken* nigdy nie jest ponownie wykorzystywany do identyfikowania innej

instancji komunikatu. Wartość *MsgToken* jest generowana przez menedżer kolejek i nie jest widoczna dla żadnej aplikacji zewnętrznej.

Jeśli komunikat jest zwracany przez wywołanie MQGET, w którym podano wersję 3 lub nowszą MQGMO, menedżer kolejek zwraca komunikat *MsgToken* identyfikujący komunikat w kolejce w MQGMO. Istnieje jeden wyjątek: jeśli komunikat jest usuwany z kolejki poza punktem synchronizacji, menedżer kolejek może nie zwrócić wartości *MsgToken* , ponieważ nie jest przydatne zidentyfikowanie zwróconego komunikatu w kolejnym wywołaniu MQGET. Aplikacje powinny odwoływać się do komunikatu w kolejnych wywołaniach MQGET tylko przy użyciu parametru *MsgToken* .

Jeśli podano parametr *MsgToken* i określono parametr *MatchOption* MQMO_MATCH_MSG_TOKEN i nie określono ani parametru MQGMO_MSG_UNDER_CURSOR, ani parametru MQGMO_BROWSE_MSG_UNDER_CURSOR, może zostać zwrócony tylko komunikat identyfikowany przez ten parametr *MsgToken* . Ta opcja jest poprawna we wszystkich kolejkach lokalnych bez względu na parametr INDXTYPE, a w systemie z/OS należy używać parametru INDXTYPE (MSGTOKEN) tylko w kolejkach menedżera obciążenia (WLM).

Wszystkie inne podane *MatchOptions* są sprawdzane i jeśli nie są zgodne, zwracany jest komunikat MQRC_NO_MSG_AVAILABLE. Jeśli komenda MQGMO_BROWSE_NEXT jest zakodowana przy użyciu znacznika MQMO_MATCH_MSG_TOKEN, komunikat identyfikowany przez *MsgToken* jest zwracany tylko wtedy, gdy znajduje się poza kursorem przeglądania dla uchwytu wywołującego.

Jeśli określono opcję MQGMO_MSG_UNDER_CURSOR lub MQGMO_BROWSE_MSG_UNDER_CURSOR, opcja MQMO_MATCH_MSG_TOKEN jest ignorowana.

Parametr MQMO_MATCH_MSG_TOKEN nie jest poprawny z następującymi opcjami pobierania komunikatów:

- MQGMO_WAIT
- MQGMO_SET_SIGNAL

W przypadku wywołania MQGET z parametrem MQMO_MATCH_MSG_TOKEN do wywołania należy dostarczyć obiekt MQGMO w wersji 3 lub nowszej, w przeciwnym razie zostanie zwrócona wartość MQRC_WRONG_GMO_VERSION.

Jeśli wartość *MsgToken* nie jest obecnie poprawna, zostanie zwrócona wartość MQCC_FAILED i MQRC_NO_MSG_AVAILABLE, chyba że wystąpi inny błąd.

ReturnedLength (MQLONG) dla MQGMO

Jest to pole wyjściowe ustawiane przez menedżer kolejek na długość danych komunikatu (w bajtach) zwracanych przez wywołanie MQGET w parametrze **Buffer** . Jeśli menedżer kolejek nie obsługuje tej możliwości, parametr *ReturnedLength* jest ustawiany na wartość MQRL_UNDEFINED.

Podczas konwersji komunikatów między kodowaniami lub zestawami znaków dane komunikatu mogą czasami zmieniać wielkość. Przy powrocie z wywołania MQGET:

- Jeśli parametr *ReturnedLength* nie ma wartości MQRL_UNDEFINED, liczba bajtów danych komunikatu zwracanych przez komendę *ReturnedLength*.
- Jeśli parametr *ReturnedLength* ma wartość MQRL_UNDEFINED, liczba bajtów zwracanych danych komunikatu jest zwykle podawana przez mniejszą z wartości *BufferLength* i *DataLength*, ale może być *mniejsza niż* , jeśli wywołanie MQGET zakończy się z kodem przyczyny MQRC_TRUNCATED_MSG_ACCEPTED. W takiej sytuacji nieistotne bajty w parametrze **Buffer** są ustawiane na wartości null.

Zdefiniowana jest następująca wartość specjalna:

MQRL_UNDEFINED (niezdefiniowana)
Nie zdefiniowano długości zwróconych danych.

W systemie z/OSwartością zwracaną w polu *ReturnedLength* jest zawsze MQRL_UNDEFINED.

Wartością początkową tego pola jest MQRL_UNDEFINED. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQGMO_VERSION_3.

Reserved2 (MQLONG) dla MQGMO

Jest to pole zastrzeżone. Wartość początkowa tego pola jest znakiem odstępu. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż **MQGMO_VERSION_4**.

MsgHandle (MQHMSG) w przypadku MQGMO

Jeśli określono opcję MQGMO_PROPERTIES_AS_Q_DEF i atrybut kolejki **PropertyControl** nie jest ustawiony na wartość MQPROP_FORCE_MQRFH2 , jest to uchwyt komunikatu, który zostanie zapełniony właściwościami komunikatu pobieranego z kolejki. Uchwyt jest tworzony przez wywołanie MQCRTMH. Wszystkie właściwości już powiązane z uchwytem zostaną wyczyszczone przed pobraniem komunikatu.

Można również podać następującą wartość:
MQHM_NONE
Nie podano uchwytu komunikatu.

Deskryptor komunikatu nie jest wymagany w wywołaniu MQGET, jeśli podano poprawny uchwyt komunikatu i został on użyty w danych wyjściowych w celu umieszczenia właściwości komunikatu. Deskryptor komunikatu powiązany z uchwytem komunikatu jest używany dla pól wejściowych.

Jeśli w wywołaniu MQGET określono deskryptor komunikatu, ma on zawsze pierwszeństwo przed deskryptorem komunikatu powiązanym z uchwytem komunikatu.

Jeśli określono wartość MQGMO_PROPERTIES_FORCE_MQRFH2 lub określono wartość MQGMO_PROPERTIES_AS_Q_DEF, a atrybut kolejki **PropertyControl** ma wartość MQPROP_FORCE_MQRFH2 , wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_MD_ERROR, jeśli nie określono parametru deskryptora komunikatu.

Po powrocie z wywołania MQGET właściwości i deskryptor komunikatu powiązane z tym uchwytem komunikatu są aktualizowane w celu odzwierciedlenia stanu pobranego komunikatu (a także deskryptora komunikatu, jeśli został on podany w wywołaniu MQGET). Właściwości komunikatu można następnie uzyskać za pomocą wywołania MQINQMP.

Z wyjątkiem rozszerzeń deskryptora komunikatu, jeśli istnieje, właściwość, do której można uzyskać dostęp za pomocą wywołania MQINQMP, nie jest zawarta w danych komunikatu. Jeśli komunikat w kolejce zawiera właściwości w danych komunikatu, są one usuwane z danych komunikatu przed zwróceniem danych do aplikacji.

Jeśli nie podano uchwytu komunikatu lub wersja jest niższa niż MQGMO_VERSION_4 , należy podać poprawny deskryptor komunikatu w wywołaniu MQGET. Wszystkie właściwości komunikatu (z wyjątkiem tych, które są zawarte w deskryptorze komunikatu) są zwracane w temacie danych komunikatu do wartości opcji właściwości w strukturze MQGMO i do atrybutu kolejki **PropertyControl** .

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQHM_NONE. To pole jest ignorowane, jeśli wartość **Version** jest mniejsza niż MQGMO_VERSION_4.

MQIIH-nagłówek informacji IMS

Struktura MQIIH opisuje informacje nagłówka dla komunikatu wysyłanego do IMS przez most IMS .W przypadku dowolnej platformy obsługiwanej przez produkt IBM MQ można utworzyć i przesłać komunikat zawierający strukturę MQIIH, ale tylko menedżer kolejek systemu IBM MQ for z/OS może używać mostu IMS . Dlatego, aby komunikat dotarły do produktu IMS z menedżera kolejek innego niżz/OS , sieć menedżera kolejek musi zawierać co najmniej jeden menedżer kolejek produktu z/OS , przez który komunikat może być kierowany.

Dostępność

Wszystkie systemy IBM MQ i klienty IBM MQ .

Nazwa formatu

MQFMT_IMS,

Zestaw znaków i kodowanie

Specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanych dla struktury MQIIH i danych komunikatu aplikacji:

- Aplikacje łączące się z menedżerem kolejek, który jest właścicielem kolejki mostu IMS , muszą udostępniać strukturę MQIIH, która jest w zestawie znaków i kodowaniu menedżera kolejek. Jest to spowodowane tym, że w tym przypadku nie jest wykonywana konwersja danych struktury MQIIH.
- Aplikacje łączące się z innymi menedżerami kolejek mogą udostępniać strukturę MQIIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań. Agent odbierającego kanału komunikatów połączony z menedżerem kolejek, który jest właścicielem kolejki mostu IMS , przekształca MQIIH.
- Dane komunikatu aplikacji po strukturze MQIIH muszą mieć ten sam zestaw znaków i kodowanie, co struktura MQIIH. Nie należy używać pól *CodedCharSetId* i *Encoding* w strukturze MQIIH do określania zestawu znaków i kodowania danych komunikatu aplikacji.

  Należy udostępnić wyjście konwersji danych w celu przekształcenia danych komunikatu aplikacji, jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 497. Pola w MQIIH dla MQIIH

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQIIH_ID_struktury	'IIH¬'
Wersja (numer wersji struktury)	MQIIH_VERSION_1	1
StrucLength (długość struktury MQIIH)	MQIIH_LENGTH_1	84
Kodowanie (zastrzeżone-patrz “Zestaw znaków i kodowanie” na stronie 417)	Brak	0
CodedCharSetId (zarezerwowany-patrz “Zestaw znaków i kodowanie” na stronie 417)	Brak	0
Format (nazwa formatu produktuMQ dla danych następujących po MQIIH)	MQFMT_BRAK	Puste
Flagi (flags)	MQIIH_BRAK	0
LTermOverride (przesłonięcie terminalu logicznego)	Brak	Puste
MFSMapName (nazwa odwzorowania usług formatu komunikatu)	Brak	Puste
ReplyToFormat (nazwa komunikatu odpowiedzi w formacieMQ)	MQFMT_BRAK	Puste
Element uwierzytelniający (RACF hasło lub przepustka)	MQIAUT_BRAK	Puste
TranInstanceId (identyfikator instancji transakcji)	MQITII_BRAK	Wartości null

Tabela 497. Pola w MQIIH dla MQIIH (kontynuacja)

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
TranState (stan transakcji)	MQITS_NOT_IN_CONVERSACJA	'¬'
CommitMode (tryb zatwierdzania)	MQICM_COMMIT_THEN_SEND	'0'
SecurityScope (zasięg zabezpieczeń)	MQISS_CHECK	'C'
Zarezerwowane (zastrzeżone)	Brak	'¬'

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQIIH_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQIIH MyIIH = {MQIIH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQIIH

```
typedef struct tagMQIIH MQIIH;
struct tagMQIIH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQIIH structure */
  MQLONG    Encoding;        /* Reserved */
  MQLONG    CodedCharSetId;  /* Reserved */
  MQCHAR8   Format;          /* MQ format name of data that follows
                                MQIIH */
  MQLONG    Flags;           /* Flags */
  MQCHAR8   LTermOverride;   /* Logical terminal override */
  MQCHAR8   MFSMapName;      /* Message format services map name */
  MQCHAR8   ReplyToFormat;   /* MQ format name of reply message */
  MQCHAR8   Authenticator;   /* RACF password or passticket */
  MQBYTE16  TranInstanceId;  /* Transaction instance identifier */
  MQCHAR    TranState;       /* Transaction state */
  MQCHAR    CommitMode;      /* Commit mode */
  MQCHAR    SecurityScope;   /* Security scope */
  MQCHAR    Reserved;        /* Reserved */
};
```

Deklaracja języka COBOL dla MQIIH

```
**   MQIIH structure
  10 MQIIH.
**    Structure identifier
   15 MQIIH-STRUCID        PIC X(4).
**    Structure version number
   15 MQIIH-VERSION        PIC S9(9) BINARY.
**    Length of MQIIH structure
   15 MQIIH-STRUCLENGTH    PIC S9(9) BINARY.
**    Reserved
   15 MQIIH-ENCODING       PIC S9(9) BINARY.
**    Reserved
   15 MQIIH-CODEDCHARSETID PIC S9(9) BINARY.
**    MQ format name of data that follows MQIIH
   15 MQIIH-FORMAT         PIC X(8).
**    Flags
   15 MQIIH-FLAGS          PIC S9(9) BINARY.
**    Logical terminal override
   15 MQIIH-LTERMOVERRIDE  PIC X(8).
```

```
**    Message format services map name
   15 MQIIH-MFSMAPNAME     PIC X(8).
**    MQ format name of reply message
   15 MQIIH-REPLYTOFORMAT  PIC X(8).
**    RACF password or passticket
   15 MQIIH-AUTHENTICATOR  PIC X(8).
**    Transaction instance identifier
   15 MQIIH-TRANINSTANCEID PIC X(16).
**    Transaction state
   15 MQIIH-TRANSTATE      PIC X.
**    Commit mode
   15 MQIIH-COMMITMODE     PIC X.
**    Security scope
   15 MQIIH-SECURITYSCOPE  PIC X.
**    Reserved
   15 MQIIH-RESERVED       PIC X.
```

Deklaracja języka PL/I dla MQIIH

```
dcl
 1 MQIIH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQIIH structure */
  3 Encoding       fixed bin(31), /* Reserved */
  3 CodedCharSetId fixed bin(31), /* Reserved */
  3 Format         char(8),       /* MQ format name of data that follows
                                     MQIIH */
  3 Flags          fixed bin(31), /* Flags */
  3 LTermOverride  char(8),       /* Logical terminal override */
  3 MFSMapName     char(8),       /* Message format services map name */
  3 ReplyToFormat  char(8),       /* MQ format name of reply message */
  3 Authenticator  char(8),       /* RACF password or passticket */
  3 TranInstanceId char(16),      /* Transaction instance identifier */
  3 TranState      char(1),       /* Transaction state */
  3 CommitMode     char(1),       /* Commit mode */
  3 SecurityScope  char(1),       /* Security scope */
  3 Reserved       char(1);       /* Reserved */
```

Deklaracja High Level Assembler dla MQIIH

```
MQIIH                 DSECT
MQIIH_STRUCID         DS   CL4   Structure identifier
MQIIH_VERSION         DS   F     Structure version number
MQIIH_STRUCLENGTH     DS   F     Length of MQIIH structure
MQIIH_ENCODING        DS   F     Reserved
MQIIH_CODEDCHARSETID  DS   F     Reserved
MQIIH_FORMAT          DS   CL8   MQ format name of data that follows
*                                MQIIH
MQIIH_FLAGS           DS   F     Flags
MQIIH_LTERMOVERRIDE   DS   CL8   Logical terminal override
MQIIH_MFSMAPNAME      DS   CL8   Message format services map name
MQIIH_REPLYTOFORMAT   DS   CL8   MQ format name of reply message
MQIIH_AUTHENTICATOR   DS   CL8   RACF password or passticket
MQIIH_TRANINSTANCEID  DS   XL16  Transaction instance identifier
MQIIH_TRANSTATE       DS   CL1   Transaction state
MQIIH_COMMITMODE      DS   CL1   Commit mode
MQIIH_SECURITYSCOPE   DS   CL1   Security scope
MQIIH_RESERVED        DS   CL1   Reserved
*
MQIIH_LENGTH          EQU  *-MQIIH
                      ORG  MQIIH
MQIIH_AREA            DS   CL(MQIIH_LENGTH)
```

Deklaracja Visual Basic dla MQIIH

```
Type MQIIH
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Length of MQIIH structure'
  Encoding       As Long     'Reserved'
  CodedCharSetId As Long     'Reserved'
  Format         As String*8 'MQ format name of data that follows MQIIH'
  Flags          As Long     'Flags'
  LTermOverride  As String*8 'Logical terminal override'
```

```
   MFSMapName     As String*8 'Message format services map name'
   ReplyToFormat  As String*8 'MQ format name of reply message'
   Authenticator  As String*8 'RACF password or passticket'
   TranInstanceId As MQBYTE16 'Transaction instance identifier'
   TranState      As String*1 'Transaction state'
   CommitMode     As String*1 'Commit mode'
   SecurityScope  As String*1 'Security scope'
   Reserved       As String*1 'Reserved'
End Type
```

StrucId (MQCHAR4) dla MQIIH

Jest to identyfikator struktury struktury nagłówka informacji IMS . Jest to zawsze pole wejściowe. Jego wartością jest MQIIH_STRUC_ID.

Wartość musi być następująca:

MQIIH_ID_struktury
Identyfikator struktury nagłówka informacji IMS .

Dla języka programowania C zdefiniowana jest również stała MQIIH_STRUC_ID_ARRAY. Ma taką samą wartość jak MQIIH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQIIH

Jest to numer wersji struktury. Wartość musi być następująca:

MQIIH_VERSION_1
Numer wersji struktury nagłówka informacji IMS .

Następująca stała określa numer wersji bieżącej:

MQIIH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka informacji IMS .

Wartością początkową tego pola jest MQIIH_VERSION_1.

StrucLength (MQLONG) dla MQIIH

Jest to długość struktury MQIIH. Wartość musi być następująca:

MQIIH_LENGTH_1
Długość struktury nagłówka informacji IMS .

Wartością początkową tego pola jest MQIIH_LENGTH_1.

Kodowanie (MQLONG) dla MQIIH

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

Kodowanie dla obsługiwanych struktur, które są zgodne ze strukturą MQIIH, jest takie samo jak kodowanie dla samej struktury MQIIH i jest pobierane z dowolnego poprzedzającego nagłówka MQ .

CodedCharSetId (MQLONG) dla MQIIH

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

Identyfikator zestawu znaków dla obsługiwanych struktur, które są zgodne ze strukturą MQIIH, jest taki sam jak identyfikator struktury MQIIH i jest pobierany z dowolnego poprzedzającego nagłówka MQ .

Format (MQCHAR8) dla MQIIH

Określa nazwę formatu MQ danych, które są zgodne ze strukturą MQIIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQIIH

Wartością opcji musi być:

MQIIH_BRAK
Brak flag.

MQIIH_PASS_EXPIRATION
Komunikat odpowiedzi zawiera:

- Te same opcje raportu utraty ważności, co w przypadku komunikatu żądania
- Pozostały czas utraty ważności z komunikatu żądania bez korekty dla czasu przetwarzania mostu

Jeśli ta wartość nie jest ustawiona, czas utraty ważności jest ustawiany na *nieograniczony*.

MQIIH_REPLY_FORMAT_NONE
Ustawia MQIIH.Format odpowiedzi na komunikat MQFMT_NONE.

MQIIH_IGNORE_PURG
Ustawia indykator TMAMIPRG w przedrostku OTMA, który żąda, aby komenda OTMA ignorowała wywołania PURG na bloku PCB TP dla transakcji CM0 .

MQIIH_CM0_REQUEST_RESPONSE
Dla transakcji w trybie kontroli transakcji 0 (CM0) ta opcja ustawia indykator TMAMHRSP w przedrostku OTMA. Ustawienie tego indykatora wymaga, aby system OTMA/IMS wygenerował komunikat DFS2082 RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY, gdy pierwotny program aplikacji IMS nie odpowiada na IOPCB ani nie przełącza się na inną transakcję.

Wartością początkową tego pola jest MQIIH_NONE.

LTermOverride (MQCHAR8) dla MQIIH

Przesłonięcie terminalu logicznego umieszczone w polu bloku PCB we/wy. Jest ona opcjonalna; jeśli nie jest określona, używana jest nazwa potoku TPIPE. Jest on ignorowany, jeśli pierwszy bajt jest pusty lub pusty.

Długość tego pola jest określona przez wartość MQ_LTERM_OVERRIDE_LENGTH. Wartość początkowa tego pola to 8 znaków odstępu.

MFSMapName (MQCHAR8) dla MQIIH

Nazwa odwzorowania usług formatu komunikatu umieszczona w polu bloku PCB we/wy. Jest ono opcjonalne. Na wejściu reprezentuje MID, na wyjściu reprezentuje MOD. Jest on ignorowany, jeśli pierwszy bajt jest pusty lub pusty.

Długość tego pola jest określona przez wartość MQ_MFS_MAP_NAME_LENGTH. Wartość początkowa tego pola to 8 znaków odstępu.

ReplyToFormat (MQCHAR8) dla MQIIH

Jest to nazwa formatu MQ komunikatu odpowiedzi, który jest wysyłany w odpowiedzi na bieżący komunikat. Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Aby przekształcić dane w komunikacie odpowiedzi za pomocą komendy MQGMO_CONVERT, należy określić wartość MQIIH.replyToFormat= MQFMT_STRING lub MQIIH.replyToFormat= MQFMT_IMS_VAR_STRING. Wyjaśnienie sposobu użycia tych pól zawiera sekcja “Format (MQCHAR8) dla deskryptora MQMD” na stronie 459.

Jeśli wartość domyślna (MQIIH.replyToFormat= MQFMT_NONE) jest używana w komunikacie żądania, a komunikat odpowiedzi jest pobierany przy użyciu komendy MQGMO_CONVERT, nie jest wykonywana żadna konwersja danych.

Element uwierzytelniający (MQCHAR8) dla MQIIH

Jest to RACF hasło lub PassTicket. Jest ona opcjonalna. Jeśli ją określono, jest ona używana z identyfikatorem użytkownika w kontekście zabezpieczeń MQMD w celu zbudowania znacznika UTOKEN, który jest wysyłany do programu IMS w celu udostępnienia kontekstu zabezpieczeń. Jeśli nie zostanie

podany, ID użytkownika zostanie użyty bez weryfikacji. Zależy to od ustawienia przełączników RACF , które mogą wymagać obecności elementu uwierzytelniającego.

Ta opcja jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL. Można użyć następującej wartości specjalnej:

MQIAUT_BRAK
Brak uwierzytelniania.

W języku programowania C zdefiniowana jest również stała MQIAUT_NONE_ARRAY. Ma ona taką samą wartość jak MQIAUT_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest określona przez parametr MQ_AUTHENTICATOR_LENGTH. Wartością początkową tego pola jest MQIAUT_NONE.

TranInstanceId (MQBYTE16) dla MQIIH

Jest to identyfikator instancji transakcji. To pole jest używane przez komunikaty wyjściowe z produktu IMS, dlatego jest ignorowane w przypadku pierwszego wejścia. Jeśli parametr *TranState* zostanie ustawiony na wartość MQITS_IN_CONVERSATION, musi on zostać podany w następnym wejściu i we wszystkich kolejnych wejściach, aby produkt IMS mógł skorelować komunikaty z poprawną konwersacją. Można użyć następującej wartości specjalnej:

MQITII_BRAK
Brak identyfikatora instancji transakcji.

W języku programowania C zdefiniowana jest również stała MQITII_NONE_ARRAY. Ma ona taką samą wartość jak MQITII_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest określona przez wartość MQ_TRAN_INSTANCE_ID_LENGTH. Wartością początkową tego pola jest MQITII_NONE.

TranState (MQCHAR) Dla MQIIH

Wskazuje stan konwersacji IMS . Jest ona ignorowana przy pierwszym wejściu, ponieważ nie istnieje żadna konwersacja. Przy kolejnych danych wejściowych wskazuje, czy konwersacja jest aktywna, czy nie. Na wyjściu jest on ustawiany przez IMS. Wartość musi być jedną z następujących wartości:

MQITS_IN_CONVERSATION
W konwersacji.

MQITS_NOT_IN_CONVERSACJA
Nie w konwersacji.

MQITS_ARCHITECTED
Zwraca dane stanu transakcji w postaci strukturalnej.

Ta wartość jest używana tylko z komendą IMS `/DISPLAY TRAN` . Zwraca dane stanu transakcji w postaci strukturalnej IMS zamiast w postaci znakowej. z/OS Więcej informacji na ten temat zawiera sekcja Pisanie programów transakcyjnych IMS za pośrednictwem produktu IBM MQ.

Wartością początkową tego pola jest MQITS_NOT_IN_CONVERSATION.

CommitMode (MQCHAR) dla MQIIH

Jest to tryb zatwierdzania IMS . Więcej informacji na temat trybów zatwierdzania IMS zawiera publikacja *OTMA Reference* . Wartość musi być jedną z następujących wartości:

MQICM_COMMIT_THEN_SEND
Zatwierdź, a następnie wyślij.

Ten tryb oznacza podwójne kolejkowanie danych wyjściowych, ale krótsze czasy zajętości regionów. Transakcje szybkie i konwersacyjne nie mogą działać w tym trybie.

MQICM_SEND_THEN_COMMIT
Wyślij, a następnie zatwierdź.

Każda transakcja IMS zainicjowana w wyniku trybu zatwierdzania MQICM_SEND_THEN_COMMIT działa w trybie odpowiedzi (RESPONSE) bez względu na sposób zdefiniowania transakcji w definicji systemu IMS (parametr MSGTYPE w makrze TRANSACT). Dotyczy to również transakcji inicjowanych za pomocą przełącznika transakcji.

Wartością początkową tego pola jest MQICM_COMMIT_THEN_SEND.

SecurityScope (MQCHAR) Dla MQIIH

Oznacza to, że wymagane jest przetwarzanie zabezpieczeń IMS . Zdefiniowane są następujące wartości:

MQISS_CHECK
Sprawdź zasięg zabezpieczeń: środowisko ACEE jest zbudowane w regionie sterującym, ale nie w regionie zależnym.

MQISS_FULL
Pełny zasięg zabezpieczeń: buforowana platforma ACEE jest budowana w regionie sterującym, a niebuforowana platforma ACEE jest budowana w regionie zależnym. Jeśli używana jest opcja MQISS_FULL, upewnij się, że ID użytkownika, dla którego zbudowano ACEE, ma dostęp do zasobów używanych w regionie zależnym.

Jeśli w tym polu nie określono wartości MQISS_CHECK ani MQISS_FULL, przyjmowana jest wartość MQISS_CHECK.

Wartością początkową tego pola jest MQISS_CHECK.

Zarezerwowany (MQCHAR) dla MQIIH

Jest to pole zastrzeżone; musi być puste.

MQIMPO-opcje właściwości zapytania o komunikat

Struktura MQIMPO umożliwia aplikacjom określanie opcji sterujących sposobem uzyskiwania informacji o właściwościach komunikatów. Struktura jest parametrem wejściowym wywołania MQINQMP.

Dostępność

Wszystkie systemy IBM MQ i klienty IBM MQ .

Zestaw znaków i kodowanie

Dane w MQIMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 498. Pola w programie MQIPMO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQIMPO_ID_struktury	'IMPO'
Wersja (numer wersji struktury)	MQIMPO_VERSION_1	1
Opcje (opcje sterujące działaniem komendy MQINQMP)	MQIMPO_INQ_FIRST	
RequestedEncoding (kodowanie, w które ma zostać przekształcona właściwość zapytania)	RODZIMA MQENC	

Tabela 498. Pola w programie MQIPMO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
RequestedCCSID (zestaw znaków właściwości zapytania)	APPL MQCCSI_PL	
ReturnedEncoding (kodowanie zwróconej wartości)	RODZIMA MQENC	
ReturnedCCSID	0	
Reserved1 (pole zastrzeżone)	znak odstępu (pole 4-bajtowe)	
ReturnedName (nazwa właściwości inquired)	MQCHARV_DEFAULT	
TypeString (reprezentacja łańcuchowa typu danych właściwości)	Pusty łańcuch lub odstępy	

Uwagi:

1. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
2. W języku programowania C: zmienna makraMQIMPO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQIMPO MyIMPO = {MQIMPO_DEFAULT};
```

Deklaracje językowe

Deklaracja języka C dla MQIMPO

```
typedef struct tagMQIMPO MQIMPO;
struct tagMQIMPO {
  MQCHAR4  StrucId;            /* Structure identifier */
  MQLONG   Version;            /* Structure version number */
  MQLONG   Options;            /* Options that control the action of
                                  MQINQMP */
  MQLONG   RequestedEncoding;  /* Requested encoding of Value */
  MQLONG   RequestedCCSID;     /* Requested character set identifier
                                  of Value */
  MQLONG   ReturnedEncoding;   /* Returned encoding of Value */
  MQLONG   ReturnedCCSID;      /* Returned character set identifier
                                  of Value */
  MQCHAR   Reserved1           /* Reserved field */
  MQCHARV  ReturnedName;       /* Returned property name */
  MQCHAR8  TypeString;         /* Property data type as a string */
};
```

Deklaracja języka COBOL dla MQIMPO

```
**   MQIMPO structure
  10 MQIMPO.
**    Structure identifier
   15 MQIMPO-STRUCID               PIC X(4).
**    Structure version number
   15 MQIMPO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQINQMP
   15 MQIMPO-OPTIONS               PIC S9(9) BINARY.
**    Requested encoding of VALUE
   15 MQIMPO-REQUESTEDENCODING     PIC S9(9) BINARY.
**    Requested character set identifier of VALUE
   15 MQIMPO-REQUESTEDCCSID        PIC S9(9) BINARY.
**    Returned encoding of VALUE
   15 MQIMPO-RETURNEDENCODING      PIC S9(9) BINARY.
**    Returned character set identifier of VALUE
```

```
   15 MQIMPO-RETURNEDCCSID          PIC S9(9) BINARY.
**    Reserved field
   15 MQIMPO-RESERVED1
**    Returned property name
   15 MQIMPO-RETURNEDNAME.
**     Address of variable length string
    20 MQIMPO-RETURNEDNAME-VSPTR    POINTER.
**     Offset of variable length string
    20 MQIMPO-RETURNEDNAME-VSOFFSET PIC S9(9) BINARY.
**     CCSID of variable length string
    20 MQIMPO-RETURNEDNAME-VSCCSID  PIC S9(9) BINARY.
**    Property data type as string
   15 MQIMPO-TYPESTRING             PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQIMPO

```
dcl
 1 MQIMPO based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 Options            fixed bin(31), /* Options that control the
                                         action of MQINQMP */
  3 RequestedEncoding  fixed bin(31), /* Requested encoding of
                                         Value */
  3 RequestedCCSID     fixed bin(31), /* Requested character set
                                         identifier of Value */
  3 ReturnedEncoding   fixed bin(31), /* Returned encoding of
                                         Value */
  3 ReturnedCCSID      fixed bin(31), /* Returned character set
                                         identifier of Value */
  3 Reserved1          fixed bin(31), /* Reserved field */
  3 ReturnedName,                     /* Returned property name */
   5 ReturnedName_VSPtr      pointer,       /* Address of returned
                                               name */
   5 5 ReturnedName_VSOffset fixed bin(31), /* Offset of returned
                                               name */
    5 5 ReturnedName_VSCCSID  fixed bin(31), /* CCSID of returned
                                               name */
  3 TypeString         char(8);       /* Property data type as
                                         string */
```

Deklaracja High Level Assembler dla MQIMPO

```
MQIMPO                       DSECT
MQIMPO_STRUCID               DS   CL4  Structure identifier
MQIMPO_VERSION               DS   F    Structure version number
MQIMPO_OPTIONS               DS   F    Options that control the
*                                      action of MQINQMP
MQIMPO_REQUESTEDENCODING     DS   F    Requested encoding of VALUE
MQIMPO_REQUESTEDCCSID        DS   F    Requested character set
*                                      identifier of VALUE
MQIMPO_RETURNEDENCODING      DS   F    Returned encoding of VALUE
MQIMPO_RETURNEDCCSID         DS   F    Returned character set
*                                      identifier of VALUE
MQIMPO_RESERVED1             DS   F    Reserved field
MQIMPO_RETURNEDNAME          DS   0F   Force fullword alignment
MQIMPO_RETURNEDNAME_VSPTR    DS   F    Address of returned name
MQIMPO_RETURNEDNAME_VSOFFSET DS   F    Offset of returned name
MQIMPO_RETURNEDNAME_VSLENGTH DS   F    Length of returned name
MQIMPO_RETURNEDNAME_VSCCSID  DS   F    CCSID of returned name
MQIMPO_RETURNEDNAME_LENGTH   EQU  *-MQIMPO_RETURNEDNAME
                             ORG  MQIMPO_RETURNEDNAME
MQIMPO_RETURNEDNAME_AREA     DS   CL(MQIMPO_RETURNEDNAME_LENGTH)
*
MQIMPO_TYPESTRING            DS   CL8  Property data type as string
MQIMPO_LENGTH                EQU  *-MQIMPO
MQIMPO_AREA                  DS   CL(MQIMPO_LENGTH)
```

StrucId (MQCHAR4) dla MQIMPO

Jest to identyfikator struktury struktury opcji właściwości komunikatu zapytania. Jest to zawsze pole wejściowe. Jego wartością jest MQIMPO_STRUC_ID.

Wartość musi być następująca:

MQIMPO_ID_struktury
Identyfikator struktury opcji właściwości komunikatu zapytania.

Dla języka programowania C zdefiniowana jest również stała MQIMPO_STRUC_ID_ARRAY. Ta wartość jest taka sama jak wartość MQIMPO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQIMPO_VERSION_1
Numer wersji struktury opcji właściwości komunikatu zapytania.

Następująca stała określa numer wersji bieżącej:

MQIMPO_CURRENT_VERSION
Bieżąca wersja struktury opcji właściwości komunikatu zapytania.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQIMPO_VERSION_1.

Opcje (MQLONG) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole Opcje

Następujące opcje sterują działaniem MQINQMP. Można określić jedną lub więcej z tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

Podano niepoprawne kombinacje opcji; wszystkie pozostałe kombinacje są poprawne.

Opcje danych wartości: Następujące opcje odnoszą się do przetwarzania danych wartości po pobraniu właściwości z komunikatu.

MQIMPO_KONVERT_VALUE
Ta opcja żąda, aby wartość właściwości została przekształcona w wartość zgodną z wartościami *RequestedCCSID* i *RequestedEncoding* określonymi przed wywołaniem MQINQMP, która zwraca wartość właściwości w obszarze *Value* .

- Jeśli konwersja zakończy się pomyślnie, pola *ReturnedCCSID* i *ReturnedEncoding* zostaną ustawione na wartość taką samą, jak pola *RequestedCCSID* i *RequestedEncoding* po powrocie z wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wartość właściwości zostanie zwrócona bez konwersji.

  Jeśli właściwość jest łańcuchem, pola *ReturnedCCSID* i *ReturnedEncoding* są ustawione na zestaw znaków i kodowanie nieprzekształconego łańcucha.

  W tym przypadku kodem zakończenia jest MQCC_WARNING z kodem przyczyny MQRC_PROP_VALUE_NOT_CONVERTED. Kursor właściwości jest przenoszony do zwróconej właściwości.

Jeśli wartość właściwości jest rozszerzana podczas konwersji i przekracza wielkość parametru **Value** , wartość jest zwracana bez konwersji z kodem zakończenia MQCC_FAILED; kod przyczyny jest ustawiony na MQRC_PROPERTY_VALUE_TOO_BIG.

Parametr **DataLength** wywołania MQINQMP zwraca długość, na jaką wartość właściwości zostałaby przekształcona, aby umożliwić aplikacji określenie wielkości buforu wymaganego do dostosowania przekształconej wartości właściwości. Kursor właściwości pozostaje niezmieniony.

Ta opcja wymaga również, aby:

- Jeśli nazwa właściwości zawiera znak wieloznaczny,
- Pole *ReturnedName* jest inicjowane z adresem lub przesunięciem dla zwróconej nazwy,

wówczas zwracana nazwa jest przekształcana w celu zapewnienia zgodności z wartościami *RequestedCCSID* i *RequestedEncoding* .

- Jeśli konwersja zakończy się pomyślnie, pole *VSCCSID* w pliku *ReturnedName* i kodowanie zwróconej nazwy zostaną ustawione na wartość wejściową *RequestedCCSID* i *RequestedEncoding*.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu lub ostrzeżenia, zwrócona nazwa nie zostanie przekształcona. W tym przypadku kodem zakończenia jest MQCC_WARNING z kodem przyczyny MQRC_PROP_NAME_NOT_CONVERTED.

  Kursor właściwości jest przenoszony do zwróconej właściwości. MQRC_PROP_VALUE_NOT_CONVERTED zwraca się, jeśli zarówno wartość, jak i nazwa nie zostały przekonwertowane.

Jeśli zwrócona nazwa zostanie rozszerzona podczas konwersji i przekroczy wielkość pola *VSBufsize* w pliku *RequestedName*, zwrócony łańcuch pozostanie nieprzekształcony, kod zakończenia MQCC_FAILED i kod przyczyny jest ustawiony na MQRC_PROPERTY_NAME_TOO_BIG.

Pole *VSLength* w strukturze MQCHARV zwraca długość, na jaką wartość właściwości zostałaby przekształcona, aby umożliwić aplikacji określenie wielkości buforu wymaganego do dostosowania przekształconej wartości właściwości. Kursor właściwości pozostaje niezmieniony.

MQIMPO_CONVERT_TYPE
Ta opcja wymaga, aby wartość właściwości została przekształcona z bieżącego typu danych w typ danych określony w parametrze **Type** wywołania MQINQMP.

- Jeśli konwersja powiedzie się, parametr **Type** pozostanie niezmieniony po zwróceniu wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wywołanie nie powiedzie się z powodu błędu MQRC_PROP_CONV_NOT_SUPPORTED. Kursor właściwości pozostaje niezmieniony.

Jeśli konwersja typu danych powoduje rozszerzenie wartości podczas konwersji, a przekształcona wartość przekracza wielkość parametru **Value** , zwracana jest wartość nieprzekształcona z kodem zakończenia MQCC_FAILED i kodem przyczyny jest MQRC_PROPERTY_VALUE_TOO_BIG.

Parametr **DataLength** wywołania MQINQMP zwraca długość, na jaką wartość właściwości zostałaby przekształcona, aby umożliwić aplikacji określenie wielkości buforu wymaganego do dostosowania przekształconej wartości właściwości. Kursor właściwości pozostaje niezmieniony.

Jeśli wartość parametru **Type** wywołania MQINQMP jest niepoprawna, wywołanie nie powiedzie się z powodu błędu MQRC_PROPERTY_TYPE_ERROR.

Jeśli żądana konwersja typu danych nie jest obsługiwana, wywołanie nie powiedzie się z powodu błędu MQRC_PROP_CONV_NOT_SUPPORTED. Obsługiwane są następujące konwersje typów danych:

Tabela 499. Obsługiwane konwersje typów danych

Typ danych właściwości	Obsługiwane docelowe typy danych
MQTYPE_BOOLEAN (wartość boolowska)	MQTYPE_STRING, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64
MQTYPE_BYTE_STRING ŁAŃCUCH	MQTYPE_STRING (łańcuch MQTYPE)
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 (łańcuch MQTYPE)
MQTYPE_FLOAT32	MQTYPE_STRING, MQTYPE_FLOAT64

Tabela 499. Obsługiwane konwersje typów danych (kontynuacja)	
Typ danych właściwości	**Obsługiwane docelowe typy danych**
MQTYPE_FLOAT64	MQTYPE_STRING (łańcuch MQTYPE)
MQTYPE_STRING (łańcuch MQTYPE)	MQTYPE_BOOLEAN, MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32, MQTYPE_INT64, MQTYPE_FLOAT32, MQTYPE_FLOAT64
MQTYPE_NULL	Brak

Ogólne zasady dotyczące obsługiwanych konwersji są następujące:

- Wartości liczbowe właściwości mogą być przekształcane z jednego typu danych na inny, pod warunkiem, że podczas konwersji nie zostaną utracone żadne dane.

  Na przykład wartość właściwości o typie danych MQTYPE_INT32 może zostać przekształcona w wartość o typie danych MQTYPE_INT64, ale nie może zostać przekształcona w wartość o typie danych MQTYPE_INT16.
- Wartość właściwości dowolnego typu danych można przekształcić w łańcuch.
- Wartość właściwości łańcuchowej może zostać przekształcona w dowolny inny typ danych, pod warunkiem, że łańcuch jest poprawnie sformatowany na potrzeby konwersji. Jeśli aplikacja próbuje przekształcić wartość właściwości łańcuchowej, która nie jest poprawnie sformatowana, program IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR.
- Jeśli aplikacja podejmie próbę konwersji, która nie jest obsługiwana, produkt IBM MQ zwróci kod przyczyny MQRC_PROP_CONV_NOT_SUPPORTED.

Konkretne reguły przekształcania wartości właściwości z jednego typu danych na inny są następujące:

- Podczas przekształcania wartości właściwości MQTYPE_BOOLEAN w łańcuch, wartość TRUE jest przekształcana w łańcuch TRUE, a wartość false jest przekształcana w łańcuch FALSE.
- Podczas przekształcania wartości właściwości MQTYPE_BOOLEAN w liczbowy typ danych wartość TRUE jest przekształcana w jeden, a wartość FALSE jest przekształcana w zero.
- Podczas przekształcania wartości właściwości łańcuchowej w wartość MQTYPE_BOOLEAN łańcuch "TRUE" lub "1" jest przekształcany w wartość TRUE, a łańcuch "FALSE" lub "0" jest przekształcany w wartość FALSE.

  Należy zauważyć, że w terminach "TRUE" i "FALSE" nie jest rozróżniana wielkość liter.

  Żaden inny łańcuch nie może zostać przekształcony; IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR.
- Podczas przekształcania wartości właściwości łańcuchowej w wartość typu danych MQTYPE_INT8, MQTYPE_INT16, MQTYPE_INT32 lub MQTYPE_INT64, łańcuch musi mieć następujący format:

```
[blanks][sign]digits
```

Znaczenia komponentów łańcucha są następujące:

blanks
Opcjonalne początkowe znaki odstępu

sign
Opcjonalny znak plus (+) lub minus (-).

digits
Ciągła sekwencja cyfr (0-9). Musi występować co najmniej jedna cyfra.

Po sekwencji cyfr łańcuch może zawierać inne znaki, które nie są cyframi, ale konwersja kończy się natychmiast po osiągnięciu pierwszego z tych znaków. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę całkowitą.

IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR, jeśli łańcuch nie jest poprawnie sformatowany.

- Podczas przekształcania wartości właściwości łańcuchowej w wartość o typie danych MQTYPE_FLOAT32 lub MQTYPE_FLOAT64, łańcuch musi mieć następujący format:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

Znaczenia komponentów łańcucha są następujące:

blanks
Opcjonalne początkowe znaki odstępu

sign
Opcjonalny znak plus (+) lub minus (-).

digits
Ciągła sekwencja cyfr (0-9). Musi występować co najmniej jedna cyfra.

e_char
Znak wykładnika, którym jest "E" lub "e".

e_sign
Opcjonalny znak plus (+) lub minus (-) dla wykładnika.

e_digits
Ciągła sekwencja cyfr (0-9) dla wykładnika. Jeśli łańcuch zawiera znak wykładnika, musi być obecny co najmniej jeden znak cyfry.

Po sekwencji cyfr lub opcjonalnych znaków reprezentujących wykładnik łańcuch może zawierać inne znaki, które nie są znakami cyfrowymi, ale konwersja kończy się natychmiast po osiągnięciu pierwszego z tych znaków. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę zmiennopozycyjną z wykładnikiem, który jest potęgą liczby 10.

IBM MQ zwraca kod przyczyny MQRC_PROP_NUMBER_FORMAT_ERROR, jeśli łańcuch nie jest poprawnie sformatowany.

- Podczas przekształcania liczbowej wartości właściwości w łańcuch, wartość jest przekształcana w łańcuchową reprezentację wartości jako liczba dziesiętna, a nie w łańcuch zawierający znak ASCII dla tej wartości. Na przykład liczba całkowita 65 jest przekształcana w łańcuch "65", a nie w łańcuch "A".
- Podczas konwersji wartości właściwości łańcucha bajtowego na łańcuch, każdy bajt jest przekształcany w dwa znaki szesnastkowe reprezentujące ten bajt. Na przykład tablica bajtów {0xF1, 0x12, 0x00, 0xFF} jest przekształcana w łańcuch "F11200FF".

DŁUGOŚĆ KOLEJKI MQIMPO_QUERY_LENGTH
Zapytanie o typ i długość wartości właściwości. Długość jest zwracana w parametrze **DataLength** wywołania MQINQMP. Wartość właściwości nie jest zwracana.

Jeśli określono bufor **ReturnedName** , pole *VSLength* struktury MQCHARV jest wypełniane długością nazwy właściwości. Nazwa właściwości nie jest zwracana.

Opcje iteracji: Następujące opcje odnoszą się do iteracji właściwości przy użyciu nazwy ze znakiem wieloznacznym

MQIMPO_INQ_FIRST
Sprawdź pierwszą właściwość, która jest zgodna z podaną nazwą. Po tym wywołaniu dla zwracanej właściwości jest ustanawiany kursor.

Jest to wartość domyślna.

Następnie można użyć opcji MQIMPO_INQ_PROP_UNDER_CURSOR z wywołaniem MQINQMP, jeśli jest to wymagane, w celu ponownego sprawdzenia tej samej właściwości.

Należy zauważyć, że istnieje tylko jeden kursor właściwości. Oznacza to, że jeśli nazwa właściwości określona w wywołaniu MQINQMP zostanie zmieniona, kursor zostanie zresetowany.

Ta opcja nie jest poprawna z żadną z następujących opcji:

MQIMPO_INQ_NEXT
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_NEXT
Pyta o następną właściwość, która jest zgodna z podaną nazwą, kontynuując wyszukiwanie od kursora właściwości. Kursor jest przenoszony do zwracanej właściwości.

Jeśli jest to pierwsze wywołanie MQINQMP dla podanej nazwy, zwracana jest pierwsza właściwość zgodna z podaną nazwą.

Opcja MQIMPO_INQ_PROP_UNDER_CURSOR może być następnie użyta z wywołaniem MQINQMP, jeśli jest to wymagane, w celu ponownego sprawdzenia tej samej właściwości.

Jeśli właściwość pod kursorem została usunięta, MQINQMP zwraca następną zgodną właściwość po tej, która została usunięta.

Jeśli zostanie dodana właściwość, która jest zgodna ze znakiem wieloznacznym, podczas gdy iteracja jest w toku, właściwość może, ale nie musi, zostać zwrócona podczas kończenia iteracji. Ta właściwość jest zwracana po zrestartowaniu iteracji za pomocą komendy MQIMPO_INQ_FIRST.

Właściwość zgodna ze znakiem wieloznacznym, który został usunięty, gdy iteracja była w toku, nie jest zwracana po jej usunięciu.

Ta opcja nie jest poprawna z żadną z następujących opcji:

MQIMPO_INQ_FIRST
MQIMPO_INQ_PROP_UNDER_CURSOR

MQIMPO_INQ_PROP_UNDER_CURSOR
Pobierz wartość właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości to ta, do której ostatnio wysłano zapytanie przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany, gdy uchwyt komunikatu jest określony w polu *MsgHandle* obiektu MQGMO w wywołaniu MQGET lub gdy uchwyt komunikatu jest określony w polach *OriginalMsgHandle* lub *NewMsgHandle* struktury MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze określony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i przyczyną jest MQRC_PROPERTY_NOT_AVAILABLE.

Ta opcja nie jest poprawna z żadną z następujących opcji:

MQIMPO_INQ_FIRST
MQIMPO_INQ_NEXT

Jeśli żadna z opisanych wcześniej opcji nie jest wymagana, można użyć następującej opcji:

MQIMPO_BRAK
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

Opcja MQIMPO_NONE jest pomocna w dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, nie można jej wykryć.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQIMPO_INQ_FIRST.

RequestedEncoding (MQLONG) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole RequestedEncoding

Jest to kodowanie, do którego ma zostać przekształcona wartość właściwości, której dotyczy zapytanie, jeśli określono wartość MQIMPO_CONVERT_VALUE lub MQIMPO_CONVERT_TYPE.

Wartością początkową tego pola jest MQENC_NATIVE.

RequestedCCSID (MQLONG) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole RequestedCCSID

Zestaw znaków, na który ma zostać przekształcona wartość właściwości zapytania, jeśli wartość jest łańcuchem znaków. Jest to także zestaw znaków, w który ma zostać przekształcony parametr *ReturnedName* , jeśli określono parametr MQIMPO_CONVERT_VALUE lub MQIMPO_CONVERT_TYPE.

Wartością początkową tego pola jest MQCCSI_APPL.

ReturnedEncoding (MQLONG) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole ReturnedEncoding

Na wyjściu jest to kodowanie zwróconej wartości.

Jeśli określono opcję MQIMPO_CONVERT_VALUE i konwersja zakończyła się pomyślnie, pole *ReturnedEncoding* przy zwracanej wartości ma taką samą wartość, jak przekazana wartość.

Wartością początkową tego pola jest MQENC_NATIVE.

ReturnedCCSID (MQLONG) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole ReturnedCCSID

W danych wyjściowych jest to zestaw znaków wartości zwracanej, jeśli parametr **Type** wywołania MQINQMP ma wartość MQTYPE_STRING.

Jeśli określono opcję MQIMPO_CONVERT_VALUE i konwersja zakończyła się pomyślnie, pole *ReturnedCCSID* przy zwracanej wartości ma taką samą wartość, jak przekazana wartość.

Wartością początkową tego pola jest zero.

Reserved1 (MQCHAR) dla MQIMPO

Jest to pole zastrzeżone. Wartością początkową tego pola jest znak odstępu (pole 4-bajtowe).

ReturnedName (MQCHARV) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole ReturnedName

Rzeczywista nazwa właściwości, do której wysłano zapytanie.

W przypadku danych wejściowych bufor łańcucha można przekazać przy użyciu pola *VSPtr* lub *VSOffset* struktury MQCHARV . Długość buforu łańcucha jest określana za pomocą pola *VSBufsize* struktury MQCHARV.

Po powrocie z wywołania MQINQMP bufor łańcucha jest uzupełniany nazwą właściwości, której dotyczy zapytanie, pod warunkiem, że bufor łańcucha był wystarczająco długi, aby w pełni zawierał nazwę. Pole *VSLength* struktury MQCHARV jest wypełniane długością nazwy właściwości. Pole *VSCCSID* struktury MQCHARV jest wypełniane w celu wskazania zestawu znaków zwracanej nazwy, niezależnie od tego, czy konwersja nazwy nie powiodła się.

Jest to pole wejściowe/wyjściowe. Wartością początkową tego pola jest MQCHARV_DEFAULT.

TypeString (MQCHAR8) dla MQIMPO
Sprawdź strukturę opcji właściwości komunikatu-pole TypeString

Reprezentacja łańcuchowa typu danych właściwości.

Jeśli właściwość została określona w nagłówku MQRFH2 i atrybut MQRFH2 `dt` nie został rozpoznany, można użyć tego pola do określenia typu danych właściwości. Kod *TypeString* jest zwracany w kodowanym zestawie znaków 1208 (UTF-8) i jest to pierwsze osiem bajtów wartości atrybutu `dt` właściwości, której rozpoznawanie nie powiodło się.

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest łańcuch pusty w języku programowania C i 8 znaków odstępu w innych językach programowania.

MQMD-deskryptor komunikatu

Struktura MQMD zawiera informacje sterujące, które towarzyszą danych aplikacji, gdy komunikat jest przesyłany między aplikacjami wysyłającymi i odbierającymi. Struktura jest parametrem wejścia/wyjścia w wywołaniach MQGET, MQPUT i MQPUT1 .

Dostępność

Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Wersja

Bieżąca wersja deskryptora MQMD to MQMD_VERSION_2. Aplikacje, które mają być przenośne między kilkoma środowiskami, muszą zapewnić, że wymagana wersja deskryptora MQMD jest obsługiwana we wszystkich odnośnych środowiskach. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję deskryptora MQMD obsługiwaną przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQMD_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *Version* numer wersji wymaganej wersji.

Dostępna jest deklaracja struktury version-1 o nazwie MQMD1.

Zestaw znaków i kodowanie

Dane w strukturze MQMD muszą być w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one podawane przez atrybut menedżera kolejek **CodedCharSetId** i parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako IBM MQ MQI client, struktura musi być w zestawie znaków i kodowaniu klienta.

Jeśli nadawcze i odbiorcze menedżery kolejek używają różnych zestawów znaków lub kodowań, dane w strukturze MQMD są przekształcane automatycznie. Przekształcanie deskryptora MQMD przez aplikację nie jest konieczne.

Korzystanie z różnych wersji deskryptora MQMD

Użycie deskryptora MQMD typu version-2 jest równoważne użyciu deskryptora MQMD typu version-1 i poprzedzenie danych komunikatu strukturą MQMDE. Jeśli jednak wszystkie pola w strukturze MQMDE mają wartości domyślne, można pominąć MQMDE. W wersji version-1 deskryptora MQMD i środowiska MQMDE są używane zgodnie z opisem:

- W wywołaniach MQPUT i MQPUT1 , jeśli aplikacja udostępnia deskryptor MQMD version-1 , aplikacja może opcjonalnie poprzedzić dane komunikatu przedrostkiem MQMDE, ustawiając pole *Format* w deskryptorze MQMD na wartość MQFMT_MD_EXTENSION w celu wskazania, że istnieje MQMDE. Jeśli aplikacja nie udostępnia środowiska MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w środowisku MQMDE.

  Uwaga: Niektóre pola, które istnieją w strukturze MQMD version-2 , ale nie w strukturze MQMD version-1 , są polami wejściowymi/wyjściowymi w wywołaniach MQPUT i MQPUT1 . Jednak menedżer kolejek nie zwraca żadnych wartości w odpowiednich polach w MQMDE na wyjściu z wywołań MQPUT i MQPUT1 . Jeśli aplikacja wymaga tych wartości wyjściowych, musi użyć deskryptora MQMD version-2 .

- W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD version-1 , menedżer kolejek dodaje przedrostek do komunikatu zwróconego z MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość inną niż domyślna. Pole *Format* w strukturze MQMD będzie miało wartość MQFMT_MD_EXTENSION wskazującą, że istnieje struktura MQMDE.

Wartości domyślne używane przez menedżer kolejek dla pól w środowisku MQMDE są takie same, jak wartości początkowe tych pól, które przedstawia Tabela 503 na stronie 487.

Jeśli komunikat znajduje się w kolejce transmisji, niektóre pola w strukturze MQMD mają ustawione określone wartości. Szczegółowe informacje na ten temat zawiera sekcja “MQXQH-nagłówek kolejki transmisji” na stronie 637 .

Kontekst komunikatu

Niektóre pola w strukturze MQMD zawierają kontekst komunikatu. Istnieją dwa typy kontekstu komunikatu: *kontekst tożsamości* i *kontekst źródłowy*. Zwykle:

- Kontekst tożsamości odnosi się do aplikacji, która *pierwotnie* umieściła komunikat.
- Kontekst źródłowy odnosi się do aplikacji, która *ostatnio* wstawiła komunikat.

Te dwie aplikacje mogą być tą samą aplikacją, ale mogą być także różnymi aplikacjami (na przykład, gdy komunikat jest przekazywany z jednej aplikacji do innej).

Chociaż kontekst tożsamości i kontekstu pochodzenia zwykle ma opisane znaczenie, treść obu typów pól kontekstu w strukturze MQMD zależy od opcji MQPMO_ * _CONTEXT, które są określone podczas umieszczania komunikatu. W związku z tym kontekst tożsamości nie musi być powiązany z aplikacją, która pierwotnie umieściła komunikat, a kontekst pochodzenia nie musi być powiązany z aplikacją, która ostatnio umieściła komunikat. Zależy to od projektu pakietu aplikacji.

Agent kanału komunikatów (MCA) nigdy nie modyfikuje kontekstu komunikatu. Atrybuty MCA, które odbierają komunikaty ze zdalnych menedżerów kolejek, używają opcji kontekstu MQPMO_SET_ALL_CONTEXT w wywołaniu MQPUT lub MQPUT1 . Dzięki temu odbierający agent MCA może zachować dokładnie kontekst komunikatu, który przebywał z komunikatem od wysyłającego agenta MCA. Jednak w wyniku tego kontekst źródłowy nie odnosi się do żadnego z agentów MCA, które wysłały i odebrały komunikat. Kontekst źródłowy odwołuje się do wcześniejszej aplikacji, która umieściła komunikat. Jeśli wszystkie aplikacje pośrednie przeszły przez kontekst komunikatu, kontekst źródłowy odwołuje się do samej aplikacji źródłowej.

W opisach pola kontekstu są opisane tak, jakby były używane w sposób opisany wcześniej. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 500. Pola w strukturze MQMD dla struktury MQMD

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQMD_STRUC_ID (Identyfikator struktury kolejki)	'MD'
Wersja (numer wersji struktury)	MQMD_VERSION_1	1
Raport (opcje dla komunikatów raportu)	MQRO_BRAK	0
MsgType (typ komunikatu)	MQMT_DATAGRAM,	8
MQMD-pole utraty ważności (czas życia komunikatu)	MQEI_UNLIMITED,	-1

Tabela 500. Pola w strukturze MQMD dla struktury MQMD (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
MQMD-pole bazy danych Feedback (informacja zwrotna lub kod przyczyny)	MQFB_BRAK	0
Kodowanie (kodowanie liczbowe danych komunikatu)	RODZIMA MQENC	Zależy od środowiska
CodedCharSetId (identyfikator zestawu znaków danych komunikatu)	MQCCSI_Q_MGR (menedżer kolejek MQ)	0
Format (nazwa formatu danych komunikatu)	MQFMT_BRAK	Puste
Priorytet (priorytet komunikatu)	MQPRI_PRIORITY_AS_Q_DEF	-1
Trwałość (trwałość komunikatu)	MQPER_PERSISTENCE_AS_Q_DEF	2
MQMD-pole MsgId (identyfikator komunikatu)	MQMI_BRAK	Wartości null
CorrelId (identyfikator korelacji)	MQCI_BRAK	Wartości null
BackoutCount (licznik wycofania)	Brak	0
ReplyToQ (nazwa kolejki odpowiedzi)	Brak	Pusty łańcuch lub odstępy
ReplyToQMgr (nazwa menedżera kolejek odpowiedzi)	Brak	Pusty łańcuch lub odstępy
UserIdentifier (identyfikator użytkownika)	Brak	Pusty łańcuch lub odstępy
AccountingToken (token rozliczania)	MQACT_NONE	Wartości null
ApplIdentityDane (dane aplikacji odnoszące się do tożsamości)	Brak	Pusty łańcuch lub odstępy
PutApplTyp (typ aplikacji, która umieściła komunikat)	MQAT_NO_CONTEXT	0
PutApplNazwa (nazwa aplikacji, która umieściła komunikat)	Brak	Pusty łańcuch lub odstępy
PutDate (data umieszczenia komunikatu)	Brak	Pusty łańcuch lub odstępy
PutTime (czas umieszczenia komunikatu)	Brak	Pusty łańcuch lub odstępy
ApplOriginData (dane aplikacji dotyczące źródła)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.		
GroupId (identyfikator grupy)	MQGI_NONE	Wartości null
MsgSeqNumber (numer kolejny komunikatu logicznego w grupie)	Brak	1
Przesunięcie (przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego)	Brak	0

Tabela 500. Pola w strukturze MQMD dla struktury MQMD (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
MQMD-pole MsgFlags (flagi komunikatu)	MQMF_BRAK	0
OriginalLength (długość oryginalnej wiadomości)	MQOL_UNDEFINED	-1

Uwagi:

1. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
2. W języku programowania C: zmienna makraMQMD_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQMD MyMD = {MQMD_DEFAULT};
```

Deklaracje językowe

Deklaracja języka C dla deskryptora MQMD

```
typedef struct tagMQMD MQMD;
struct tagMQMD {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    Report;            /* Options for report messages */
  MQLONG    MsgType;           /* Message type */
  MQLONG    Expiry;            /* Message lifetime */
  MQLONG    Feedback;          /* Feedback or reason code */
  MQLONG    Encoding;          /* Numeric encoding of message data */
  MQLONG    CodedCharSetId;    /* Character set identifier of message
                                  data */
  MQCHAR8   Format;            /* Format name of message data */
  MQLONG    Priority;          /* Message priority */
  MQLONG    Persistence;       /* Message persistence */
  MQBYTE24  MsgId;             /* Message identifier */
  MQBYTE24  CorrelId;          /* Correlation identifier */
  MQLONG    BackoutCount;      /* Backout counter */
  MQCHAR48  ReplyToQ;          /* Name of reply queue */
  MQCHAR48  ReplyToQMgr;       /* Name of reply queue manager */
  MQCHAR12  UserIdentifier;    /* User identifier */
  MQBYTE32  AccountingToken;   /* Accounting token */
  MQCHAR32  ApplIdentityData;  /* Application data relating to
                                  identity */
  MQLONG    PutApplType;       /* Type of application that put the
                                  message */
  MQCHAR28  PutApplName;       /* Name of application that put the
                                  message */
  MQCHAR8   PutDate;           /* Date when message was put */
  MQCHAR8   PutTime;           /* Time when message was put */
  MQCHAR4   ApplOriginData;    /* Application data relating to origin */
  MQBYTE24  GroupId;           /* Group identifier */
  MQLONG    MsgSeqNumber;      /* Sequence number of logical message
                                  within group */
  MQLONG    Offset;            /* Offset of data in physical message
                                  from start of logical message */
  MQLONG    MsgFlags;          /* Message flags */
  MQLONG    OriginalLength;    /* Length of original message */
};
```

Deklaracja języka COBOL dla deskryptora MQMD

```
**   MQMD structure
  10 MQMD.
**    Structure identifier
   15 MQMD-STRUCID          PIC X(4).
**    Structure version number
   15 MQMD-VERSION          PIC S9(9) BINARY.
```

```
**    Options for report messages
   15 MQMD-REPORT           PIC S9(9) BINARY.
**    Message type
   15 MQMD-MSGTYPE          PIC S9(9) BINARY.
**    Message lifetime
   15 MQMD-EXPIRY           PIC S9(9) BINARY.
**    Feedback or reason code
   15 MQMD-FEEDBACK         PIC S9(9) BINARY.
**    Numeric encoding of message data
   15 MQMD-ENCODING         PIC S9(9) BINARY.
**    Character set identifier of message data
   15 MQMD-CODEDCHARSETID   PIC S9(9) BINARY.
**    Format name of message data
   15 MQMD-FORMAT           PIC X(8).
**    Message priority
   15 MQMD-PRIORITY         PIC S9(9) BINARY.
**    Message persistence
   15 MQMD-PERSISTENCE      PIC S9(9) BINARY.
**    Message identifier
   15 MQMD-MSGID            PIC X(24).
**    Correlation identifier
   15 MQMD-CORRELID         PIC X(24).
**    Backout counter
   15 MQMD-BACKOUTCOUNT     PIC S9(9) BINARY.
**    Name of reply queue
   15 MQMD-REPLYTOQ         PIC X(48).
**    Name of reply queue manager
   15 MQMD-REPLYTOQMGR      PIC X(48).
**    User identifier
   15 MQMD-USERIDENTIFIER   PIC X(12).
**    Accounting token
   15 MQMD-ACCOUNTINGTOKEN  PIC X(32).
**    Application data relating to identity
   15 MQMD-APPLIDENTITYDATA PIC X(32).
**    Type of application that put the message
   15 MQMD-PUTAPPLTYPE      PIC S9(9) BINARY.
**    Name of application that put the message
   15 MQMD-PUTAPPLNAME      PIC X(28).
**    Date when message was put
   15 MQMD-PUTDATE          PIC X(8).
**    Time when message was put
   15 MQMD-PUTTIME          PIC X(8).
**    Application data relating to origin
   15 MQMD-APPLORIGINDATA   PIC X(4).
**    Group identifier
   15 MQMD-GROUPID          PIC X(24).
**    Sequence number of logical message within group
   15 MQMD-MSGSEQNUMBER     PIC S9(9) BINARY.
**    Offset of data in physical message from start of logical message
   15 MQMD-OFFSET           PIC S9(9) BINARY.
**    Message flags
   15 MQMD-MSGFLAGS         PIC S9(9) BINARY.
**    Length of original message
   15 MQMD-ORIGINALLENGTH   PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla deskryptora MQMD

```
dcl
 1 MQMD based,
  3 StrucId          char(4),       /* Structure identifier */
  3 Version          fixed bin(31), /* Structure version number */
  3 Report           fixed bin(31), /* Options for report messages */
  3 MsgType          fixed bin(31), /* Message type */
  3 Expiry           fixed bin(31), /* Message lifetime */
  3 Feedback         fixed bin(31), /* Feedback or reason code */
  3 Encoding         fixed bin(31), /* Numeric encoding of message
                                       data */
  3 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                       message data */
  3 Format           char(8),       /* Format name of message data */
  3 Priority         fixed bin(31), /* Message priority */
  3 Persistence      fixed bin(31), /* Message persistence */
  3 MsgId            char(24),      /* Message identifier */
  3 CorrelId         char(24),      /* Correlation identifier */
  3 BackoutCount     fixed bin(31), /* Backout counter */
  3 ReplyToQ         char(48),      /* Name of reply queue */
  3 ReplyToQMgr      char(48),      /* Name of reply queue manager */
  3 UserIdentifier   char(12),      /* User identifier */
  3 AccountingToken  char(32),      /* Accounting token */
```

```
  3 ApplIdentityData char(32),      /* Application data relating to
                                       identity */
  3 PutApplType      fixed bin(31), /* Type of application that put the
                                       message */
  3 PutApplName      char(28),      /* Name of application that put the
                                       message */
  3 PutDate          char(8),       /* Date when message was put */
  3 PutTime          char(8),       /* Time when message was put */
  3 ApplOriginData   char(4),       /* Application data relating to
                                       origin */
  3 GroupId          char(24),      /* Group identifier */
  3 MsgSeqNumber     fixed bin(31), /* Sequence number of logical
                                       message within group */
  3 Offset           fixed bin(31), /* Offset of data in physical
                                       message from start of logical
                                       message */
  3 MsgFlags         fixed bin(31), /* Message flags */
  3 OriginalLength   fixed bin(31); /* Length of original message */
```

Deklaracja High Level Assembler dla deskryptora MQMD

```
MQMD                    DSECT
MQMD_STRUCID            DS   CL4   Structure identifier
MQMD_VERSION            DS   F     Structure version number
MQMD_REPORT             DS   F     Options for report messages
MQMD_MSGTYPE            DS   F     Message type
MQMD_EXPIRY             DS   F     Message lifetime
MQMD_FEEDBACK           DS   F     Feedback or reason code
MQMD_ENCODING           DS   F     Numeric encoding of message data
MQMD_CODEDCHARSETID     DS   F     Character set identifier of message
*                                  data
MQMD_FORMAT             DS   CL8   Format name of message data
MQMD_PRIORITY           DS   F     Message priority
MQMD_PERSISTENCE        DS   F     Message persistence
MQMD_MSGID              DS   XL24  Message identifier
MQMD_CORRELID           DS   XL24  Correlation identifier
MQMD_BACKOUTCOUNT       DS   F     Backout counter
MQMD_REPLYTOQ           DS   CL48  Name of reply queue
MQMD_REPLYTOQMGR        DS   CL48  Name of reply queue manager
MQMD_USERIDENTIFIER     DS   CL12  User identifier
MQMD_ACCOUNTINGTOKEN    DS   XL32  Accounting token
MQMD_APPLIDENTITYDATA   DS   CL32  Application data relating to identity
MQMD_PUTAPPLTYPE        DS   F     Type of application that put the
*                                  message
MQMD_PUTAPPLNAME        DS   CL28  Name of application that put the
*                                  message
MQMD_PUTDATE            DS   CL8   Date when message was put
MQMD_PUTTIME            DS   CL8   Time when message was put
MQMD_APPLORIGINDATA     DS   CL4   Application data relating to origin
MQMD_GROUPID            DS   XL24  Group identifier
MQMD_MSGSEQNUMBER       DS   F     Sequence number of logical message
*                                  within group
MQMD_OFFSET             DS   F     Offset of data in physical message
*                                  from start of logical message
MQMD_MSGFLAGS           DS   F     Message flags
MQMD_ORIGINALLENGTH     DS   F     Length of original message
*
MQMD_LENGTH             EQU  *-MQMD
                        ORG  MQMD
MQMD_AREA               DS   CL(MQMD_LENGTH)
```

Deklaracja Visual Basic dla MQMD

```
Type MQMD
  StrucId          As String*4  'Structure identifier'
  Version          As Long      'Structure version number'
  Report           As Long      'Options for report messages'
  MsgType          As Long      'Message type'
  Expiry           As Long      'Message lifetime'
  Feedback         As Long      'Feedback or reason code'
  Encoding         As Long      'Numeric encoding of message data'
  CodedCharSetId   As Long      'Character set identifier of message'
                                'data'
  Format           As String*8  'Format name of message data'
  Priority         As Long      'Message priority'
  Persistence      As Long      'Message persistence'
  MsgId            As MQBYTE24  'Message identifier'
```

```
   CorrelId         As MQBYTE24  'Correlation identifier'
   BackoutCount     As Long      'Backout counter'
   ReplyToQ         As String*48 'Name of reply queue'
   ReplyToQMgr      As String*48 'Name of reply queue manager'
   UserIdentifier   As String*12 'User identifier'
   AccountingToken  As MQBYTE32  'Accounting token'
   ApplIdentityData As String*32 'Application data relating to identity'
   PutApplType      As Long      'Type of application that put the'
                                 'message'
   PutApplName      As String*28 'Name of application that put the'
                                 'message'
   PutDate          As String*8  'Date when message was put'
   PutTime          As String*8  'Time when message was put'
   ApplOriginData   As String*4  'Application data relating to origin'
   GroupId          As MQBYTE24  'Group identifier'
   MsgSeqNumber     As Long      'Sequence number of logical message'
                                 'within group'
   Offset           As Long      'Offset of data in physical message'
                                 'from start of logical message'
   MsgFlags         As Long      'Message flags'
   OriginalLength   As Long      'Length of original message'
End Type
```

StrucId (MQCHAR4) dla MQMD

Jest to identyfikator struktury struktury deskryptora komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQMD_STRUC_ID.

Wartość musi być następująca:

MQMD_STRUC_ID (Identyfikator struktury kolejki)
Identyfikator struktury deskryptora komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQMD_STRUC_ID_ARRAY. Ma taką samą wartość jak MQMD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQMD

Jest to numer wersji struktury i musi mieć jedną z następujących wartości:

MQMD_VERSION_1
Struktura deskryptora komunikatu Version-1 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQMD_VERSION_2
Struktura deskryptora komunikatu Version-2 .

Ta wersja jest obsługiwana we wszystkich środowiskach IBM MQ V6.0 i nowszych oraz IBM MQ MQI clients połączonych z tymi systemami.

Uwaga: Jeśli używana jest struktura MQMD w wersji version-2 , menedżer kolejek wykonuje dodatkowe sprawdzenia dla wszystkich struktur nagłówków MQ , które mogą być obecne na początku danych komunikatu aplikacji. Szczegółowe informacje można znaleźć w uwagach dotyczących składni wywołania MQPUT.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQMD_CURRENT_VERSION
Bieżąca wersja struktury deskryptora komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQMD_VERSION_1.

Raport (MQLONG) dla MQMD

Komunikat raportu to komunikat o innym komunikacie, używany do informowania aplikacji o oczekiwanych lub nieoczekiwanych zdarzeniach, które odnoszą się do oryginalnego komunikatu. Pole *Report* umożliwia aplikacji wysyłającej oryginalny komunikat w celu określenia, które komunikaty raportu są wymagane, czy dane komunikatu aplikacji mają być w nich uwzględnione, a także (zarówno w przypadku raportów, jak i odpowiedzi) w jaki sposób mają zostać ustawione identyfikatory komunikatu

i korelacji w komunikacie raportu lub odpowiedzi. Można zażądać dowolnego lub wszystkich (lub żadnego) następujących typów komunikatów raportu:

- Wyjątek
- Termin ważności
- Potwierdź po przybyciu (COA)
- Potwierdź przy dostawie (COD)
- Powiadomienie o działaniu pozytywnym (PAN)
- Powiadomienie o działaniu negatywnym (NAN)

Można określić jedną lub więcej z tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

Aplikacja odbierająca komunikat raportu może określić przyczynę wygenerowania raportu, sprawdzając pole *Feedback* w strukturze MQMD. Więcej szczegółów zawiera pole *Feedback* .

Użycie opcji raportu podczas umieszczania komunikatu w temacie może spowodować wygenerowanie i wysłanie do aplikacji zera, jednego lub wielu komunikatów raportu. Jest to spowodowane tym, że komunikat publikacji może zostać wysłany do zero, jednej lub wielu aplikacji subskrybujących.

Opcje wyjątku: Podaj jedną z wymienionych opcji, aby zażądać komunikatu raportu wyjątku.

MQRO_EXCEPTION,
Agent kanału komunikatów generuje ten typ raportu, gdy komunikat jest wysyłany do innego menedżera kolejek i nie może zostać dostarczony do określonej kolejki docelowej. Na przykład kolejka docelowa lub pośrednia kolejka transmisji może być pełna lub komunikat może być zbyt duży dla kolejki.

Generowanie komunikatu raportu o wyjątku zależy od trwałości oryginalnego komunikatu i szybkości kanału komunikatów (normalnego lub szybkiego), przez który przechodzi oryginalny komunikat:

- Dla wszystkich trwałych komunikatów oraz dla nietrwałych komunikatów przechodujących przez normalne kanały komunikatów raport o wyjątkach jest generowany tylko wtedy, gdy działanie określone przez aplikację wysyłającą dla warunku błędu może zostać zakończone pomyślnie. Aplikacja wysyłająca może określić jedną z następujących czynności w celu sterowania rozdysponowaniem oryginalnego komunikatu w przypadku wystąpienia warunku błędu:
  - MQRO_DEAD_LETTER_Q (powoduje umieszczenie oryginalnego komunikatu w kolejce niedostarczonych komunikatów).
  - MQRO_DISCARD_MSG (to powoduje usunięcie oryginalnego komunikatu).

  Jeśli działanie określone przez aplikację wysyłającą nie może zostać zakończone pomyślnie, oryginalny komunikat pozostaje w kolejce transmisji i nie jest generowany żaden komunikat raportu o wyjątku.
- W przypadku nietrwałych komunikatów przechodzącego przez kanały szybkich komunikatów oryginalny komunikat jest usuwany z kolejki transmisji i generowany jest raport o wyjątkach *nawet wtedy* , gdy określone działanie dla warunku błędu nie może zostać zakończone pomyślnie. Jeśli na przykład określono parametr MQRO_DEAD_LETTER_Q, ale nie można umieścić oryginalnego komunikatu w kolejce niedostarczonych komunikatów, ponieważ ta kolejka jest pełna, zostanie wygenerowany komunikat raportu o wyjątku i oryginalny komunikat zostanie odrzucony.

  Więcej informacji na temat normalnych i szybkich kanałów komunikatów zawiera sekcja Szybkość komunikatów nietrwałych (NPMSPEED).

Raport o wyjątku nie jest generowany, jeśli aplikacja, która umieściła oryginalny komunikat, może zostać powiadomiona synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 .

Aplikacje mogą również wysyłać raporty o wyjątkach w celu wskazania, że komunikat nie może zostać przetworzony (na przykład dlatego, że jest to transakcja debetowa, która spowodowałaby przekroczenie limitu kredytowego konta).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy podawać więcej niż jednej wartości MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA i MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_DATA
Jest to taka sama sytuacja jak w przypadku parametru MQRO_EXCEPTION, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest dołączanych do komunikatu raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA i MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_EXCEPTION_WITH_FULL_DATA
Raporty wyjątków z wymaganymi pełnymi danymi.

Jest on taki sam jak parametr MQRO_EXCEPTION, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości MQRO_EXCEPTION, MQRO_EXCEPTION_WITH_DATA i MQRO_EXCEPTION_WITH_FULL_DATA.

Opcje utraty ważności: Podaj jedną z wymienionych opcji, aby zażądać komunikatu raportu o utracie ważności.

MQRO_UTR_WAŻN.
Ten typ raportu jest generowany przez menedżer kolejek, jeśli komunikat zostanie odrzucony przed dostarczeniem do aplikacji, ponieważ upłynął jego czas ważności (patrz pole *Expiry*). Jeśli ta opcja nie jest ustawiona, komunikat raportu nie jest generowany, jeśli komunikat zostanie odrzucony z tego powodu (nawet jeśli określono jedną z opcji MQRO_EXCEPTION_ *).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy określać więcej niż jednej wartości MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA i MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_DATA
Jest taka sama jak MQRO_EXPIRATION, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu zostanie dołączonych do komunikatu raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Nie należy określać więcej niż jednej wartości MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA i MQRO_EXPIRATION_WITH_FULL_DATA.

MQRO_EXPIRATION_WITH_FULL_DATA
Jest to takie samo jak MQRO_EXPIRATION, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu zostaną dołączone do komunikatu raportu.

Nie należy określać więcej niż jednej wartości MQRO_EXPIRATION, MQRO_EXPIRATION_WITH_DATA i MQRO_EXPIRATION_WITH_FULL_DATA.

Opcje potwierdzania przy odbiorze: Podaj jedną z wymienionych opcji, aby zażądać komunikatu raportu potwierdzania przy odbiorze.

MQRO_COA

Ten typ raportu jest generowany przez menedżera kolejek, który jest właścicielem kolejki docelowej, gdy komunikat jest umieszczany w kolejce docelowej. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest umieszczany jako część jednostki pracy, a kolejka docelowa jest kolejką lokalną, komunikat raportu COA wygenerowany przez menedżer kolejek może zostać odtworzony tylko wtedy, gdy jednostka pracy została zatwierdzona.

Raport COA nie jest generowany, jeśli pole *Format* w deskryptorze komunikatu to MQFMT_XMIT_Q_HEADER lub MQFMT_DEAD_LETTER_HEADER. Zapobiega to generowaniu raportu COA, jeśli komunikat jest umieszczany w kolejce transmisji lub nie można go dostarczyć i umieścić w kolejce niedostarczonych komunikatów.

W przypadku kolejki mostu IMS raport COA jest generowany, gdy komunikat dotrze do kolejki IMS (potwierdzenie odebrane z serwisu IMS). i nie wtedy, gdy komunikat jest umieszczany w kolejce mostu MQ . Oznacza to, że jeśli parametr IMS nie jest aktywny, raport COA nie jest generowany do momentu uruchomienia programu IMS i umieszczenia komunikatu w kolejce IMS .

Użytkownik uruchamiający program, który umieszcza komunikat w programie MQMD.Report= MQRO_COA musi mieć uprawnienie + passid w kolejce odpowiedzi. Jeśli użytkownik nie ma uprawnienia + passid, komunikat raportu COA nie dociera do kolejki odpowiedzi. Podjęto próbę umieszczenia komunikatu raportu w kolejce niedostarczonych komunikatów.

Nie należy podawać więcej niż jednej wartości MQRO_COA, MQRO_COA_WITH_DATA i MQRO_COA_WITH_FULL_DATA.

MQRO_KOA_WITH_DATA
Jest ona taka sama jak MQRO_COA, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest uwzględnianych w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości MQRO_COA, MQRO_COA_WITH_DATA i MQRO_COA_WITH_FULL_DATA.

MQRO_KOA_WITH_FULL_DATA
Jest ona taka sama jak MQRO_COA, z tym wyjątkiem, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości MQRO_COA, MQRO_COA_WITH_DATA i MQRO_COA_WITH_FULL_DATA.

Opcje potwierdzania przy dostarczeniu: Podaj jedną z wymienionych opcji, aby zażądać komunikatu raportu potwierdzania przy dostarczeniu.

MQRO_COD
Ten typ raportu jest generowany przez menedżer kolejek, gdy aplikacja pobiera komunikat z kolejki docelowej w sposób usuwający komunikat z kolejki. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest wczytywany jako część jednostki pracy, komunikat raportu jest generowany w ramach tej samej jednostki pracy, dzięki czemu raport nie jest dostępny do momentu zatwierdzenia jednostki pracy. Jeśli jednostka pracy zostanie wycofana, raport nie zostanie wysłany.

Raport COD nie jest zawsze generowany, jeśli komunikat jest pobierany z opcją MQGMO_MARK_SKIP_BACKOUT. Jeśli podstawowa jednostka pracy zostanie wycofana, ale dodatkowa jednostka pracy zostanie zatwierdzona, komunikat zostanie usunięty z kolejki, ale raport COD nie zostanie wygenerowany.

Raport COD nie jest generowany, jeśli pole *Format* w deskryptorze komunikatu ma wartość MQFMT_DEAD_LETTER_HEADER. Zapobiega to generowaniu raportu COD, jeśli nie można dostarczyć komunikatu i jest on umieszczany w kolejce niedostarczonych komunikatów.

Opcja MQRO_COD nie jest poprawna, jeśli kolejka docelowa jest kolejką XCF.

Nie należy podawać więcej niż jednej wartości MQRO_COD, MQRO_COD_WITH_DATA i MQRO_COD_WITH_FULL_DATA.

MQRO_KOD_WITH_DATA
Jest taka sama jak opcja MQRO_COD, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest uwzględnianych w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Jeśli parametr MQGMO_ACCEPT_TRUNCATED_MSG zostanie określony w wywołaniu MQGET dla oryginalnego komunikatu, a pobrany komunikat zostanie obcięty, ilość danych komunikatu aplikacji umieszczanych w komunikacie raportu zależy od środowiska:

- W systemie z/OSjest to minimum z następujących wartości:
  - Długość oryginalnego komunikatu
  - Długość buforu używanego do pobierania komunikatu
  - 100 bajtów.
- W innych środowiskach jest to minimum z następujących wartości:
  - Długość oryginalnego komunikatu
  - 100 bajtów.

Parametr MQRO_COD_WITH_DATA jest niepoprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie należy podawać więcej niż jednej wartości MQRO_COD, MQRO_COD_WITH_DATA i MQRO_COD_WITH_FULL_DATA.

MQRO_KOD_WITH_FULL_DATA
Jest taka sama jak opcja MQRO_COD, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są uwzględniane w komunikacie raportu.

Parametr MQRO_COD_WITH_FULL_DATA jest niepoprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie należy podawać więcej niż jednej wartości MQRO_COD, MQRO_COD_WITH_DATA i MQRO_COD_WITH_FULL_DATA.

Opcje powiadomienia o działaniu: Podaj jedną lub obie wymienione opcje, aby zażądać od aplikacji odbierającej wysłania komunikatu raportu o działaniu pozytywnym lub negatywnym.

MQRO_PAN
Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Wskazuje, że działanie żądane w komunikacie zostało wykonane pomyślnie. Aplikacja generująca raport określa, czy do raportu mają zostać dołączone jakiekolwiek dane.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. W razie potrzeby raport musi zostać wygenerowany przez aplikację pobierającą.

MQRO_NAN
Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Oznacza to, że działanie żądane w komunikacie nie zostało wykonane pomyślnie. Aplikacja generująca raport określa, czy do raportu mają zostać dołączone jakiekolwiek dane. Na przykład można dołączyć dane wskazujące, dlaczego żądanie nie mogło zostać wykonane.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. W razie potrzeby raport musi zostać wygenerowany przez aplikację pobierającą.

Wniosek musi określić, które warunki odpowiadają działaniu pozytywnemu i które odpowiadają działaniu negatywnemu. Jeśli jednak żądanie zostało wykonane tylko częściowo, należy na żądanie wygenerować raport NAN, a nie raport PAN. Każdy możliwy warunek musi odpowiadać działaniu pozytywnemu lub negatywnemu, ale nie obu.

Opcje identyfikatora komunikatu: Określ jedną z wymienionych opcji, aby sterować sposobem ustawiania *MsgId* komunikatu raportu (lub komunikatu odpowiedzi).

MQRO_NEW_MSG_ID
Jest to działanie domyślne i wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, dla raportu lub komunikatu odpowiedzi zostanie wygenerowana nowa wartość *MsgId* .

MQRO_PASS_MSG_ID
Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, plik *MsgId* tego komunikatu jest kopiowany do pliku *MsgId* komunikatu raportu lub odpowiedzi.

Wartość *MsgId* komunikatu publikacji będzie różna dla każdego subskrybenta, który otrzymuje kopię publikacji, i dlatego wartość *MsgId* skopiowana do komunikatu raportu lub odpowiedzi będzie różna dla każdego subskrybenta.

Jeśli ta opcja nie zostanie podana, zostanie przyjęty identyfikator MQRO_NEW_MSG_ID.

Opcje identyfikatora korelacji: należy podać jedną z wymienionych opcji, aby sterować sposobem ustawiania *CorrelId* komunikatu raportu (lub komunikatu odpowiedzi).

MQRO_COPY_MSG_ID_TO_CORREL_ID
Jest to działanie domyślne i wskazuje, że jeśli raport lub odpowiedź zostanie wygenerowana w wyniku tego komunikatu, *MsgId* tego komunikatu zostanie skopiowany do *CorrelId* komunikatu raportu lub odpowiedzi.

Wartość *MsgId* komunikatu publikacji będzie inna dla każdego subskrybenta, który otrzymuje kopię publikacji i dlatego wartość *MsgId* skopiowana do pliku *CorrelId* komunikatu raportu lub odpowiedzi będzie inna dla każdego z nich.

MQRO_PASS_CORREL_ID (Identyfikator KORELACJI MQ)
Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, plik *CorrelId* tego komunikatu jest kopiowany do pliku *CorrelId* komunikatu raportu lub odpowiedzi.

Identyfikator *CorrelId* komunikatu publikacji będzie specyficzny dla subskrybenta, chyba że używa on opcji MQSO_SET_CORREL_ID i ustawia pole identyfikatora SubCorrelw pliku MQSD na wartość MQCI_NONE. Dlatego możliwe jest, że *CorrelId* skopiowany do *CorrelId* raportu lub komunikatu odpowiedzi będzie inny dla każdego z nich.

Jeśli ta opcja nie zostanie podana, zostanie przyjęty identyfikator MQRO_COPY_MSG_ID_TO_CORREL_ID.

Serwery odpowiadające na żądania lub generujące komunikaty raportu muszą sprawdzić, czy opcje MQRO_PASS_MSG_ID lub MQRO_PASS_CORREL_ID zostały ustawione w oryginalnym komunikacie. Jeśli tak, serwery muszą wykonać opisane działanie dla tych opcji. Jeśli żadna z tych opcji nie jest ustawiona, serwery muszą wykonać odpowiednie działanie domyślne.

Opcje rozporządzania: Określ jedną z wymienionych opcji, aby kontrolować rozporządzanie oryginalnym komunikatem, gdy nie może on zostać dostarczony do kolejki docelowej. Aplikacja może ustawić opcje rozporządzania niezależnie od żądań raportów o wyjątkach.

MQRO_DEAD_LETTER_Q
Jest to działanie domyślne i umieszcza komunikat w kolejce niedostarczonych komunikatów, jeśli nie można dostarczyć komunikatu do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieściła oryginalny komunikat, nie może być synchronicznie powiadamiana o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 . Jeśli nadawca zażądał komunikatu o wyjątku, zostanie on wygenerowany.
- Gdy aplikacja, która wstawiła oryginalny komunikat, umieszczała go w temacie.

MQRO_DISCARD_MSG
Spowoduje to usunięcie komunikatu, jeśli nie można go dostarczyć do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieściła oryginalny komunikat, nie może być synchronicznie powiadamiana o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 . Jeśli nadawca zażądał komunikatu o wyjątku, zostanie on wygenerowany.
- Gdy aplikacja, która wstawiła oryginalny komunikat, umieszczała go w temacie.

Jeśli pierwotny komunikat ma zostać zwrócony do nadawcy bez umieszczania oryginalnego komunikatu w kolejce niedostarczonych komunikatów, nadawca musi określić parametr MQRO_DISCARD_MSG z wartością MQRO_EXCEPTION_WITH_FULL_DATA.

MQRO_PASS_DISCARD_AND_WAŻNOŚCI
Jeśli ta opcja jest ustawiona dla komunikatu, a raport lub odpowiedź jest generowana z jego powodu, deskryptor komunikatu raportu dziedziczy:

- MQRO_DISCARD_MSG, jeśli został ustawiony.
- Pozostały czas utraty ważności komunikatu (jeśli nie jest to raport dotyczący utraty ważności). Jeśli jest to raport dotyczący utraty ważności, czas utraty ważności jest ustawiany na 60 sekund.

Opcja działania

DZIAŁANIE MQRO_ACTIVITY
Użycie tej wartości umożliwia śledzenie trasy **dowolnego** komunikatu w sieci menedżera kolejek. Opcja raportu może być określona dla każdego komunikatu bieżącego użytkownika, co pozwala na natychmiastowe rozpoczęcie obliczania trasy komunikatu przez sieć.

Jeśli aplikacja generująca komunikat nie może włączyć generowania raportu o aktywności, raportowanie można włączyć przy użyciu wyjścia przecięcia funkcji API udostępnianego przez administratorów menedżera kolejek.

Uwaga:

1. Im mniej menedżerów kolejek w sieci jest w stanie generować raporty aktywności, tym mniej szczegółowa trasa.
2. Raporty aktywności mogą być trudne do umieszczenia we właściwej kolejności w celu określenia podjętej trasy.
3. Raporty aktywności mogą nie być w stanie znaleźć trasy do żądanego miejsca docelowego.
4. Komunikaty z tym zestawem opcji raportu muszą zostać zaakceptowane przez dowolny menedżer kolejek, nawet jeśli nie rozumie on tej opcji. Umożliwia to ustawienie opcji raportu dla dowolnego komunikatu użytkownika, nawet jeśli są one przetwarzane przez menedżer kolejek w wersji innej niżIBM WebSphere MQ 6.0 lub nowszej.
5. Jeśli proces (menedżer kolejek lub proces użytkownika) wykonuje działanie na komunikacie z tą opcją, może on wybrać generowanie i umieszczanie raportu działań.

Opcja domyślna: Jeśli nie są wymagane żadne opcje raportu, należy podać następujące informacje:

MQRO_BRAK
Ta wartość wskazuje, że nie określono żadnych innych opcji. Parametr MQRO_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Informacje ogólne:

1. Wszystkie wymagane typy raportów muszą być specjalnie żądane przez aplikację wysyłającą oryginalny komunikat. Na przykład, jeśli zażądano raportu COA, ale raport o wyjątku nie został wygenerowany, raport COA jest generowany, gdy komunikat jest umieszczany w kolejce docelowej, ale nie jest generowany raport o wyjątku, jeśli kolejka docelowa jest pełna w momencie nadejścia komunikatu. Jeśli nie ustawiono opcji *Report* , nie są generowane żadne komunikaty raportu przez menedżer kolejek ani agent kanału komunikatów (MCA).

   Niektóre opcje raportu można określić nawet wtedy, gdy lokalny menedżer kolejek ich nie rozpoznaje. Jest to przydatne, gdy opcja ma być przetwarzana przez *docelowy* menedżer kolejek. Więcej informacji na temat zawiera sekcja “Opcje raportu i flagi komunikatów” na stronie 933.

   Jeśli żądany jest komunikat raportu, w polu *ReplyToQ* należy podać nazwę kolejki, do której ma zostać wysłany raport. Po odebraniu komunikatu raportu rodzaj raportu można określić, sprawdzając pole *Feedback* w deskryptorze komunikatu.
2. Jeśli menedżer kolejek lub agent MCA generujący komunikat raportu nie może umieścić komunikatu raportu w kolejce odpowiedzi (na przykład z powodu zapełnienia kolejki odpowiedzi lub kolejki transmisji), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów. Jeśli to *również* nie powiedzie się lub nie ma kolejki niedostarczonych komunikatów, podjęte działanie zależy od typu komunikatu raportu:

- Jeśli komunikat raportu jest raportem o wyjątkach, komunikat, który wygenerował raport o wyjątkach, pozostaje w kolejce transmisji, co zapewnia, że komunikat nie zostanie utracony.
- W przypadku wszystkich innych typów raportów komunikat raportu jest odrzucany i przetwarzanie jest kontynuowane normalnie. Dzieje się tak, ponieważ oryginalny komunikat został już dostarczony bezpiecznie (dla komunikatów raportu COA lub COD) lub nie jest już interesujący (dla komunikatu raportu o utracie ważności).

Po pomyślnym umieszczeniu komunikatu raportu w kolejce (kolejka docelowa lub pośrednia kolejka transmisji) komunikat nie podlega już specjalnemu przetwarzaniu; jest traktowany tak samo, jak każdy inny komunikat.

3. Po wygenerowaniu raportu otwierana jest kolejka *ReplyToQ* i umieszczany jest komunikat raportu z uprawnieniami *UserIdentifier* w deskryptorze MQMD komunikatu będącego przyczyną raportu, z wyjątkiem następujących przypadków:
   - Raporty o wyjątkach generowane przez odbierający agent MCA są umieszczane z uprawnieniami używanymi przez agent MCA podczas próby umieszczenia komunikatu powodującego wygenerowanie raportu.
   - Raporty COA wygenerowane przez menedżera kolejek są umieszczane z uprawnieniami używanymi podczas umieszczania komunikatu powodującego umieszczenie raportu w menedżerze kolejek generującym raport. Jeśli na przykład komunikat został umieszczony przez odbierającego agenta MCA przy użyciu identyfikatora użytkownika agenta MCA, menedżer kolejek umieszcza raport COA przy użyciu identyfikatora użytkownika agenta MCA.

   Aplikacje generujące raporty muszą używać tych samych uprawnień, które są używane do generowania odpowiedzi. Jest to zwykle uprawnienie identyfikatora użytkownika w oryginalnym komunikacie.

   Jeśli raport musi być przemieszczany do zdalnego miejsca docelowego, nadawcy i odbiorcy mogą zdecydować, czy go zaakceptować, w taki sam sposób, jak w przypadku innych komunikatów.
4. Jeśli żądany jest komunikat raportu z danymi:
   - Komunikat raportu jest zawsze generowany z ilością danych żądanych przez nadawcę oryginalnego komunikatu. Jeśli komunikat raportu jest zbyt duży dla kolejki odpowiedzi, następuje przetwarzanie opisane powyżej; komunikat raportu nigdy nie jest obcinany, aby zmieścił się w kolejce odpowiedzi.
   - Jeśli *Format* oryginalnego komunikatu to MQFMT_XMIT_Q_HEADER, dane zawarte w raporcie nie obejmują MQXQH. Dane raportu rozpoczynają się od pierwszego bajtu danych poza MQXQH w oryginalnym komunikacie. Dzieje się tak niezależnie od tego, czy kolejka jest kolejką transmisji.
5. Jeśli w kolejce odpowiedzi zostanie odebrany komunikat COA, COD lub raportu o utracie ważności, to jest pewne, że oryginalny komunikat został odebrany, dostarczony lub utracił ważność (w zależności od przypadku). Jeśli jednak co najmniej jeden z tych komunikatów raportu zostanie zażądany i nie zostanie odebrany, nie można przyjąć wartości odwrotnej, ponieważ mogła wystąpić jedna z następujących sytuacji:
   a. Komunikat raportu jest wstrzymany, ponieważ odsyłacz jest wyłączony.
   b. Komunikat raportu jest wstrzymany, ponieważ warunek blokowania istnieje w pośredniej kolejce transmisji lub w kolejce odpowiedzi (na przykład kolejka jest pełna lub zablokowana dla operacji umieszczania).
   c. Komunikat raportu znajduje się w kolejce niedostarczonych komunikatów.
   d. Gdy menedżer kolejek próbował wygenerować komunikat raportu, nie mógł umieścić go w odpowiedniej kolejce ani w kolejce niedostarczonych komunikatów, dlatego nie można wygenerować komunikatu raportu.
   e. Wystąpiło niepowodzenie menedżera kolejek między zgłoszonym działaniem (nadejście, dostarczenie lub utrata ważności) a wygenerowaniem odpowiedniego komunikatu raportu. (Nie dzieje się tak w przypadku komunikatów raportu COD, jeśli aplikacja pobiera oryginalny komunikat w jednostce pracy, ponieważ komunikat raportu COD jest generowany w tej samej jednostce pracy).

   Komunikaty raportów o wyjątkach mogą być wstrzymane w ten sam sposób z powodów 1, 2 i 3 powyżej. Jeśli jednak agent MCA nie może wygenerować komunikatu raportu o wyjątku (komunikat raportu nie może zostać umieszczony ani w kolejce odpowiedzi, ani w kolejce niedostarczonych

komunikatów), oryginalny komunikat pozostaje w kolejce transmisji u nadawcy, a kanał jest zamknięty. Ma to miejsce bez względu na to, czy komunikat raportu miał zostać wygenerowany na wysyłającym, czy odbierającym końcu kanału.

6. Jeśli pierwotny komunikat jest tymczasowo zablokowany (co powoduje wygenerowanie komunikatu raportu o wyjątku i umieszczenie oryginalnego komunikatu w kolejce niedostarczonych komunikatów), ale blokada zostanie skasowana, a aplikacja odczyta oryginalny komunikat z kolejki niedostarczonych komunikatów i umieści go ponownie w miejscu docelowym, mogą wystąpić następujące warunki:
   - Mimo że został wygenerowany komunikat raportu o wyjątku, oryginalny komunikat w końcu dotarł pomyślnie do miejsca docelowego.
   - W odniesieniu do pojedynczego oryginalnego komunikatu generowany jest więcej niż jeden komunikat raportu o wyjątku, ponieważ oryginalny komunikat może napotkać później inną blokadę.

Zgłaszanie komunikatów podczas umieszczania w temacie:

1. Raporty mogą być generowane podczas umieszczania komunikatu w temacie. Ten komunikat zostanie wysłany do wszystkich subskrybentów tematu, który może mieć wartość zero, jeden lub wiele. Należy to wziąć pod uwagę przy wyborze opcji raportu, ponieważ w wyniku tego może zostać wygenerowanych wiele komunikatów raportu.
2. Podczas umieszczania komunikatu w temacie może istnieć wiele kolejek docelowych, którym ma zostać nadana kopia komunikatu. Jeśli w niektórych z tych kolejek docelowych występuje problem, taki jak zapełnienie kolejki, pomyślne zakończenie operacji MQPUT zależy od ustawienia parametru NPMSGDLV lub PMSGDLV (w zależności od trwałości komunikatu). Jeśli ustawienie jest takie, że dostarczenie komunikatu do kolejki docelowej musi zakończyć się pomyślnie (na przykład jest to komunikat trwały do trwałego subskrybenta, a parametr PMSGDLV jest ustawiony na wartość ALL lub ALLDUR), to powodzenie jest definiowane jako jedno z następujących kryteriów:
   - Pomyślnie wstawiono do kolejki subskrybenta
   - Użycie opcji MQRO_DEAD_LETTER_Q i pomyślne umieszczenie w kolejce niedostarczonych komunikatów, jeśli kolejka subskrybenta nie może pobrać komunikatu
   - Użycie MQRO_DISCARD_MSG, jeśli kolejka subskrybenta nie może pobrać komunikatu.

Raportowanie komunikatów dla segmentów komunikatów:

1. Komunikaty raportu mogą być żądane dla komunikatów, dla których dozwolona jest segmentacja (patrz opis flagi MQMF_SEGMENTATION_ALLOWED). Jeśli menedżer kolejek uzna, że konieczne jest podzielenie komunikatu na segmenty, może zostać wygenerowany komunikat raportu dla każdego z segmentów, który następnie napotka odpowiedni warunek. Aplikacje muszą być przygotowane do odbierania wielu komunikatów raportu dla każdego typu żądanego komunikatu raportu. Użyj pola *GroupId* w komunikacie raportu, aby skorelować wiele raportów z identyfikatorem grupy oryginalnego komunikatu, a pole *Feedback* określa typ każdego komunikatu raportu.
2. Jeśli parametr MQGMO_LOGICAL_ORDER jest używany do pobierania komunikatów raportu dla segmentów, należy pamiętać, że kolejne wywołania MQGET mogą zwracać raporty *różnych typów* . Jeśli na przykład zarówno raporty COA, jak i COD są żądane dla komunikatu segmentowanego przez menedżer kolejek, wywołania MQGET dla komunikatów raportu mogą zwrócić komunikaty raportu COA i COD przeplatające się w nieprzewidywalny sposób. Aby tego uniknąć, należy użyć opcji MQGMO_COMPLETE_MSG (opcjonalnie z opcją MQGMO_ACCEPT_TRUNCATED_MSG). MQGMO_COMPLETE_MSG powoduje, że menedżer kolejek składa ponownie komunikaty raportu, które mają ten sam typ raportu. Na przykład pierwsze wywołanie MQGET może ponownie złożyć wszystkie komunikaty COA związane z pierwotnym komunikatem, a drugie wywołanie MQGET może ponownie złożyć wszystkie komunikaty COD. To, który komunikat jest składany jako pierwszy, zależy od typu komunikatu raportu występującego jako pierwszy w kolejce.
3. Aplikacje, które same umieszczają segmenty, mogą określać różne opcje raportu dla każdego segmentu. Należy jednak zwrócić uwagę na następujące kwestie:
   - Jeśli segmenty są pobierane przy użyciu opcji MQGMO_COMPLETE_MSG, tylko opcje raportu w *pierwszym* segmencie są uznawane przez menedżer kolejek.

- Jeśli segmenty są pobierane jeden po drugim, a większość z nich ma jedną z opcji MQRO_COD_ *, ale co najmniej jeden segment nie, nie można użyć opcji MQGMO_COMPLETE_MSG do pobrania komunikatów raportu z pojedynczym wywołaniem MQGET lub użyć opcji MQGMO_ALL_SEGMENTS_AVAILABLE, aby wykryć, kiedy wszystkie komunikaty raportu nadeszły.

4. W sieci produktu MQ menedżery kolejek mogą mieć różne możliwości. Jeśli komunikat raportu dla segmentu jest generowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji, menedżer kolejek lub agent MCA domyślnie nie dołączają niezbędnych informacji o segmencie do komunikatu raportu, co może utrudnić zidentyfikowanie oryginalnego komunikatu, który spowodował wygenerowanie raportu. Aby uniknąć tej trudności, należy wysłać żądanie danych z komunikatem raportu, to znaczy podając odpowiednie opcje MQRO_ * _WITH_DATA lub MQRO_ * _WITH_FULL_DATA. Należy jednak pamiętać, że jeśli określono parametr MQRO_ * _WITH_DATA, *mniej niż* 100 bajtów danych komunikatu aplikacji może zostać zwróconych do aplikacji, która pobiera komunikat raportu, jeśli komunikat raportu został wygenerowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji.

Treść deskryptora komunikatu dla komunikatu raportu: gdy menedżer kolejek lub agent kanału komunikatów (MCA) generuje komunikat raportu, ustawia pola w deskryptorze komunikatu na następujące wartości, a następnie umieszcza komunikat w normalny sposób.

Tabela 501. Wartości używane w polach MQMD, gdy komunikat raportu jest generowany przez system

Pole w strukturze MQMD	Użyta wartość
StrucId	MQMD_STRUC_ID (Identyfikator struktury kolejki)
Version	MQMD_VERSION_2
Report	MQRO_BRAK
MsgType	MQMT_RAPORT
Expiry	MQEI_UNLIMITED,
Feedback	Odpowiednie dla rodzaju raportu (MQFB_COA, MQFB_COD, MQFB_EXPIRATION lub wartość MQRC_ *)
Encoding	Skopiowane z oryginalnego deskryptora komunikatu
CodedCharSetId	Skopiowane z oryginalnego deskryptora komunikatu
Format	Skopiowane z oryginalnego deskryptora komunikatu
Priority	Skopiowane z oryginalnego deskryptora komunikatu
Persistence	Skopiowane z oryginalnego deskryptora komunikatu
MsgId	Zgodnie z opcjami raportu w oryginalnym deskryptorze komunikatu
CorrelId	Zgodnie z opcjami raportu w oryginalnym deskryptorze komunikatu
BackoutCount	0
ReplyToQ	Puste
ReplyToQMgr	Nazwa menedżera kolejek.
UserIdentifier	Zgodnie z opcją MQPMO_PASS_IDENTITY_CONTEXT
AccountingToken	Zgodnie z opcją MQPMO_PASS_IDENTITY_CONTEXT
ApplIdentityData	Zgodnie z opcją MQPMO_PASS_IDENTITY_CONTEXT
PutApplType	MQAT_QMGR lub odpowiednio dla agenta kanału komunikatów
PutApplName	Pierwsze 28 bajtów nazwy menedżera kolejek lub nazwy agenta kanału komunikatów. W przypadku komunikatów raportu wygenerowanych przez most IMS to pole zawiera nazwę grupy XCF i nazwę elementu XCF systemu IMS , do którego odnosi się komunikat.

Tabela 501. Wartości używane w polach MQMD, gdy komunikat raportu jest generowany przez system (kontynuacja)

Pole w strukturze MQMD	Użyta wartość
PutDate	Data wysłania komunikatu raportu
PutTime	Czas wysłania komunikatu raportu
ApplOriginData	Puste
GroupId	Skopiowane z oryginalnego deskryptora komunikatu
MsgSeqNumber	Skopiowane z oryginalnego deskryptora komunikatu
Offset	Skopiowane z oryginalnego deskryptora komunikatu
MsgFlags	Skopiowane z oryginalnego deskryptora komunikatu
OriginalLength	Kopiowany z oryginalnego deskryptora komunikatu, jeśli nie jest określony parametr MQOL_UNDEFINED, i ustawiany na długość oryginalnych danych komunikatu. W przeciwnym razie

Zaleca się, aby aplikacja generująca raport ustawiła podobne wartości, z wyjątkiem następujących:

- Pole *ReplyToQMgr* można ustawić na wartość pustą (menedżer kolejek zmienia tę wartość na nazwę lokalnego menedżera kolejek, gdy komunikat jest umieszczany).
- Ustaw pola kontekstu przy użyciu opcji, która zostałaby użyta dla odpowiedzi, zwykle MQPMO_PASS_IDENTITY_CONTEXT.

Analizowanie pola raportu: Pole *Report* zawiera podpola. Z tego powodu aplikacje, które muszą sprawdzić, czy nadawca komunikatu, którego dotyczy żądanie, musi użyć jednej z technik opisanych w sekcji "Analizowanie pola raportu" na stronie 935.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQRO_NONE.

MsgType (MQLONG) w deskryptorze MQMD

Wskazuje typ komunikatu. Typy komunikatów są pogrupowane w następujący sposób:

MQMT_SYSTEM_FIRST
Najniższa wartość dla typów komunikatów zdefiniowanych przez system.

MQMT_SYSTEM_LAST
Najwyższa wartość dla typów komunikatów zdefiniowanych przez system.

Następujące wartości są obecnie zdefiniowane w zakresie systemowym:

MQMT_DATAGRAM,
Komunikat nie wymaga odpowiedzi.

MQMT_ŻĄDANIE
Komunikat wymaga odpowiedzi.

W polu *ReplyToQ* podaj nazwę kolejki, do której ma zostać wysłana odpowiedź. Pole *Report* wskazuje sposób ustawiania wartości *MsgId* i *CorrelId* odpowiedzi.

MQMT_REPLY
Komunikat jest odpowiedzią na wcześniejszy komunikat żądania (MQMT_REQUEST). Komunikat musi zostać wysłany do kolejki wskazanej w polu *ReplyToQ* komunikatu żądania. Pole *Report* żądania służy do sterowania sposobem ustawiania wartości *MsgId* i *CorrelId* odpowiedzi.

Uwaga: Menedżer kolejek nie wymusza relacji żądanie-odpowiedź; jest to odpowiedzialność za aplikację.

MQMT_RAPORT
Komunikat jest raportowany w przypadku nieoczekiwanego lub oczekiwanego wystąpienia, zwykle związanego z innym komunikatem (na przykład odebrano komunikat żądania, który zawierał niepoprawne dane). Wyślij komunikat do kolejki wskazanej w polu *ReplyToQ* deskryptora komunikatu oryginalnego komunikatu. Ustaw pola *Feedback* , aby wskazywały rodzaj raportu. Pole *Report* oryginalnego komunikatu służy do sterowania sposobem ustawiania wartości *MsgId* i *CorrelId* komunikatu raportu.

Komunikaty raportów wygenerowane przez menedżera kolejek lub agenta kanału komunikatów są zawsze wysyłane do kolejki *ReplyToQ* z polami *Feedback* i *CorrelId* ustawionymi zgodnie z powyższym opisem.

Można również użyć wartości zdefiniowanych przez aplikację. Muszą one mieścić się w następującym zakresie:

MQMT_APPL_FIRST
Najniższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

MQMT_APPL_LAST
Najwyższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

W przypadku wywołań MQPUT i MQPUT1 wartość *MsgType* musi być w zakresie zdefiniowanym przez system lub w zakresie zdefiniowanym przez aplikację. Jeśli nie, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MSG_TYPE_ERROR.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQMT_DATAGRAM.

Utrata ważności (MQLONG) dla deskryptora MQMD

Jest to okres czasu wyrażony w dziesiątych częściach sekundy, ustawiony przez aplikację, która umieszcza komunikat. Komunikat zostanie zakwalifikowany do usunięcia, jeśli nie został usunięty z kolejki docelowej przed upływem tego czasu.

Na przykład, aby ustawić czas utraty ważności na jedną minutę, należy ustawić wartość **MQMD**.**Expiry** do 600.

Wartość jest zmniejszana w celu odzwierciedlenia czasu, jaki komunikat spędza w kolejce docelowej, a także w pośrednich kolejkach transmisji, jeśli operacja umieszczania jest w kolejce zdalnej. Może być również zmniejszana przez agenty kanału komunikatów w celu odzwierciedlenia czasów transmisji, jeśli są one istotne. Podobnie aplikacja przekazująca ten komunikat do innej kolejki może w razie potrzeby zmniejszyć tę wartość, jeśli zachowała komunikat przez dłuższy czas. Jednak czas ważności jest traktowany jako przybliżony, a wartość nie musi być zmniejszana w celu odzwierciedlenia niewielkich przedziałów czasu.

Gdy komunikat jest pobierany przez aplikację przy użyciu wywołania MQGET, pole *Expiry* reprezentuje pozostały czas ważności.

Po upływie czasu utraty ważności komunikatu zostaje on zakwalifikowany do usunięcia przez menedżer kolejek. Komunikat jest odrzucany, gdy wystąpi wywołanie MQGET przeglądania lub nieprzeglądania, które zwróciłoby komunikat, gdyby nie utracił on ważności. Na przykład nieprzeglądanie wywołania MQGET z polem *MatchOptions* w MQGMO ustawionym na wartość MQMO_NONE podczas odczytu z kolejki uporządkowanej FIFO powoduje usunięcie wszystkich komunikatów, które utraciły ważność, aż do pierwszego komunikatu, który nie utracił ważności. W przypadku kolejki uporządkowanej według priorytetu to samo wywołanie spowoduje usunięcie komunikatów o wyższym priorytecie, które utraciły ważność, oraz komunikatów o takim samym priorytecie, które dotarły do kolejki przed pierwszym komunikatem, który nie utracił ważności.

Komunikat, który utracił ważność, nigdy nie jest zwracany do aplikacji (ani przez przeglądanie, ani przez wywołanie MQGET bez przeglądania), więc wartość w polu *Expiry* deskryptora komunikatu po pomyślnym wywołaniu MQGET jest większa niż zero lub wartość specjalna MQEI_UNLIMITED.

Jeśli komunikat jest umieszczany w kolejce zdalnej, może utracić ważność (i zostać odrzucony), gdy znajduje się w pośredniej kolejce transmisji, zanim komunikat dotrze do kolejki docelowej.

Raport jest generowany po odrzucaniu komunikatu, który utracił ważność, jeśli w komunikacie określono jedną z opcji raportu MQRO_EXPIRATION_ *. Jeśli nie zostanie podana żadna z tych opcji, nie zostanie wygenerowany taki raport. Komunikat nie będzie już istotny po upływie tego okresu (być może dlatego, że został zastąpiony przez późniejszy komunikat).

W przypadku komunikatu umieszczonego w punkcie synchronizacji odstęp czasu utraty ważności rozpoczyna się w momencie umieszczenia komunikatu, a nie w momencie zatwierdzenia punktu synchronizacji. Możliwe, że interwał utraty ważności może upłynąć przed zatwierdzeniem punktu synchronizacji. W takim przypadku komunikat zostanie odrzucony w pewnym momencie po operacji zatwierdzania, a komunikat nie zostanie zwrócony do aplikacji w odpowiedzi na operację MQGET.

Każdy inny program, który usuwa komunikaty na podstawie czasu utraty ważności, musi również wysłać odpowiedni komunikat raportu, jeśli został zażądany.

Uwagi:

1. Jeśli komunikat zostanie umieszczony z czasem *Expiry* równym zero lub liczbą większą od 999 999 999, wywołanie MQPUT lub MQPUT1 nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_EXPIRY_ERROR; w tym przypadku nie zostanie wygenerowany żaden komunikat raportu.

   Aby włączyć kod przyczyny 2013, MQRC_EXPIRY_ERROR, należy włączyć zmienną środowiskową AMQ_ENFORCE_MAX_EXPIRY_ERROR.

   W poniższym przykładzie użyto parametru Linux:

   ```
   $ export AMQ_ENFORCE_MAX_EXPIRY_ERROR=True
   ```

   Należy zauważyć, że:

   - Ważne jest, aby wyeksportować zmienną
   - Wartość rzeczywista jest ignorowana, jednak użycie parametru `True` może być pomocne podczas przeglądania konfiguracji.
2. Ponieważ komunikat z czasem utraty ważności, który upłynął, może nie zostać odrzucony do czasu późniejszego, w kolejce mogą znajdować się komunikaty, które przekroczyli swój czas utraty ważności i dlatego nie mogą zostać pobrane. Te komunikaty są jednak uwzględniane w liczbie komunikatów w kolejce dla wszystkich celów, włącznie z wyzwalaniem głębokości.

   Jeśli subskrybent/konsument (klient) próbuje uzyskać komunikat, który utracił ważność, klient nie otrzymuje nic, ponieważ komunikat został odrzucony, ponieważ był zbyt stary. Ponadto klient nie otrzyma żadnego komunikatu o błędzie.
3. Raport o utracie ważności jest generowany, jeśli zostanie zgłoszony, gdy komunikat zostanie odrzucony, a nie wtedy, gdy zostanie zakwalifikowany do usunięcia.
4. Usunięcie komunikatu, który utracił ważność, i wygenerowanie raportu o utracie ważności (jeśli jest to wymagane) nigdy nie jest częścią jednostki pracy aplikacji, nawet jeśli komunikat został zaplanowany do usunięcia w wyniku wywołania MQGET działającego w ramach jednostki pracy.
5. Jeśli prawie wygasły komunikat jest pobierany przez wywołanie MQGET w obrębie jednostki pracy, a następnie jednostka pracy jest wycofywana, komunikat może zostać zakwalifikowany do usunięcia przed ponownym pobraniem.
6. Jeśli prawie wygasły komunikat jest zablokowany przez wywołanie MQGET z opcją MQGMO_LOCK, komunikat może zostać zakwalifikowany do usunięcia przed pobraniem go przez wywołanie MQGET z opcją MQGMO_MSG_UNDER_CURSOR; kod przyczyny MQRC_NO_MSG_UNDER_CURSOR jest zwracany w przypadku tego kolejnego wywołania MQGET.
7. Po pobraniu komunikatu żądania z czasem utraty ważności większym niż zero aplikacja może wykonać jedno z następujących działań podczas wysyłania komunikatu odpowiedzi:

   - Skopiuj pozostały czas ważności z komunikatu żądania do komunikatu odpowiedzi.
   - Ustaw czas utraty ważności w komunikacie odpowiedzi na wartość jawną większą niż zero.
   - Ustaw czas utraty ważności w komunikacie odpowiedzi na wartość MQEI_UNLIMITED.

Działanie, które należy wykonać, zależy od projektu aplikacji. Jednak domyślne działanie dotyczące umieszczania komunikatów w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) musi zachowywać pozostały czas utraty ważności komunikatu i kontynuować jego zmniejszanie.

8. Komunikaty wyzwalacza są zawsze generowane z wartością MQEI_UNLIMITED.
9. Komunikat (zwykle w kolejce transmisji) o nazwie *Format* MQFMT_XMIT_Q_HEADER ma drugi deskryptor komunikatu w MQXQH. Dlatego jest z nim powiązane dwa pola *Expiry* . W tym przypadku należy zwrócić uwagę na następujące dodatkowe kwestie:
   - Gdy aplikacja umieszcza komunikat w kolejce zdalnej, menedżer kolejek umieszcza komunikat początkowo w lokalnej kolejce transmisji i poprzedza dane komunikatu aplikacji strukturą MQXQH. Menedżer kolejek ustawia wartości dwóch pól *Expiry* na takie same, jak te określone przez aplikację.

     Jeśli aplikacja umieszcza komunikat bezpośrednio w lokalnej kolejce transmisji, dane komunikatu muszą już zaczynać się od struktury MQXQH, a nazwą formatu musi być MQFMT_XMIT_Q_HEADER. W takim przypadku aplikacja nie musi ustawiać jednakowe wartości tych dwóch pól *Expiry* . (Menedżer kolejek sprawdza, czy pole *Expiry* w MQXQH zawiera poprawną wartość i czy dane komunikatu są wystarczająco długie, aby je dołączyć). W przypadku aplikacji, która może zapisywać bezpośrednio do kolejki transmisji, aplikacja musi utworzyć nagłówek kolejki transmisji z osadzonym deskryptorem komunikatu. Jeśli jednak wartość utraty ważności w deskryptorze komunikatu zapisanym w kolejce transmisji jest niespójna z wartością w osadzonym deskryptorze komunikatu, występuje odrzucenie błędu utraty ważności.
   - Gdy komunikat o nazwie *Format* MQFMT_XMIT_Q_HEADER jest pobierany z kolejki (bez względu na to, czy jest to kolejka normalna, czy kolejka transmisji), menedżer kolejek zmniejsza *oba* te *Expiry* pola wraz z czasem oczekiwania w kolejce. Jeśli dane komunikatu nie są wystarczająco długie, aby uwzględnić pole *Expiry* w MQXQH, nie jest zgłaszany żaden błąd.
   - Menedżer kolejek używa pola *Expiry* w osobnym deskryptorze komunikatu (czyli nie w deskryptorze komunikatu osadzonym w strukturze MQXQH) do sprawdzenia, czy komunikat kwalifikuje się do usunięcia.
   - Jeśli wartości początkowe dwóch pól *Expiry* są różne, czas *Expiry* w osobnym deskryptorze komunikatu podczas pobierania komunikatu może być większy niż zero (dlatego komunikat nie jest zakwalifikowany do usunięcia), podczas gdy upłynął czas zgodny z wartością pola *Expiry* w MQXQH. W tym przypadku pole *Expiry* w MQXQH jest ustawione na wartość zero.
10. Czas utraty ważności komunikatu odpowiedzi zwróconego z mostu IMS jest nieograniczony, chyba że w polu Flags w MQIIH jest ustawiona wartość MQIIH_PASS_EXPIRATION. Więcej informacji na ten temat zawiera sekcja Flagi .

Rozpoznawana jest następująca wartość specjalna:

MQEI_UNLIMITED,
Komunikat ma nieograniczony czas ważności.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQEI_UNLIMITED.

Komunikaty, które utraciły ważność w systemie z/OS
W systemie IBM MQ for z/OSkomunikaty, które utraciły ważność, są usuwane przez następne odpowiednie wywołanie MQGET.

Jeśli jednak takie wywołanie nie wystąpi, komunikat, który utracił ważność, nie jest usuwany, a w przypadku niektórych kolejek może zostać zgromadzona duża liczba komunikatów, które utraciły ważność. Aby rozwiązać ten problem, należy ustawić dla menedżera kolejek okresowe skanowanie kolejek i usuwanie komunikatów, które utraciły ważność w jednej lub kilku kolejkach, w jeden z następujących sposobów:

Skanowanie okresowe

Okres można określić za pomocą atrybutu menedżera kolejek EXPRYINT (interwał utraty ważności). Za każdym razem, gdy upłynie okres ważności, menedżer kolejek wyszukuje kolejki kandydujące, które są warte skanowania w celu usunięcia komunikatów, które utraciły ważność.

Menedżer kolejek przechowuje w każdej kolejce informacje o komunikatach, które utraciły ważność, i wie, czy warto przejrzeć komunikaty, które utraciły ważność. Tak więc w dowolnym momencie skanowany jest tylko wybór kolejek.

Kolejki współużytkowane są skanowane tylko przez jednego menedżera kolejek w grupie współużytkowania kolejek. Zwykle jest to pierwszy menedżer kolejek, który ma zostać zrestartowany, lub pierwszy, który ma ustawiony parametr EXPRYINT. Jeśli ten menedżer kolejek zostanie zakończony, skanowanie kolejki zostanie przejęte przez innego menedżera kolejek w grupie współużytkowania kolejek. Ustaw wartość okresu ważności dla wszystkich menedżerów kolejek w grupie współużytkowania kolejek na tę samą wartość.

Należy zauważyć, że przetwarzanie utraty ważności jest wykonywane dla każdej kolejki podczas restartowania menedżera kolejek, niezależnie od ustawienia EXPRYINT.

Jawne żądanie

Wydaj komendę REFRESH QMGR TYPE (TERMIN WAŻNOŚCI), podając kolejkę lub kolejki, które mają być skanowane.

Wymuszanie niższych czasów ważności

Administratorzy mogą ograniczyć czas utraty ważności każdego komunikatu umieszczonego w kolejce lub temacie, używając atrybutu **CAPEXPRY** określonego w atrybucie **CUSTOM** w kolejce lub temacie.

Ważne: z/OS Nie można używać atrybutu **CAPEXPRY** wprowadzonego w pliku IBM MQ 9.3.1 z klastrem, jeśli pełne repozytorium znajduje się w systemie z/OS.

Multi V 9.3.1 W produkcie IBM MQ 9.3.1administratorzy mogą ograniczyć czas utraty ważności za pomocą atrybutu **CAPEXPRY** kolejki lub tematu bez konieczności podawania go w atrybucie **CUSTOM** . Jeśli dla kolejki lub tematu ustawiono już atrybut **CAPEXPRY** w atrybucie **CUSTOM** , należy usunąć atrybut **CAPEXPRY** z atrybutu **CUSTOM** przed zmodyfikowaniem nowego atrybutu **CAPEXPRY** . Można to zrobić za pomocą jednej komendy, na przykład:

```
ALTER QLOCAL(Q1) CAPEXPRY(1000) CUSTOM('')
```

Uwaga: Podczas migrowania obiektu z wcześniejszej wersji produktu wartość **CAPEXPRY** jest ustawiana na wartość domyślną NOLIMIT. Jeśli atrybut **CAPEXPRY** został ustawiony wewnątrz atrybutu **CUSTOM** , ta opcja ma pierwszeństwo.

Aby użyć nowego atrybutu **CAPEXPRY** , należy najpierw usunąć atrybut **CAPEXPRY** z atrybutu **CUSTOM** . Ustawienie obu opcji nie działa.

Multi V 9.3.1 Zmodyfikowany atrybut **CAPEXPRY** ustawiony bezpośrednio w kolejkach lub tematach (poza atrybutem **CUSTOM**) jest atrybutem klastrowym. Należy zauważyć, że wszystkie instancje kolejki klastra powinny używać tej samej wartości dla swojego atrybutu **CAPEXPRY** . Nadal możliwe jest, że kolejka transmisji skróci czas utraty ważności komunikatu, jeśli w kolejce transmisji ustawiono parametr **CAPEXPRY** , a jego wartość jest mniejsza niż wartość atrybutu **CAPEXPRY** kolejki klastra.

Czas utraty ważności określony w polu **Expiry** deskryptora MQMD przez aplikację, który jest większy niż wartość parametru **CAPEXPRY** określona w kolejce lub temacie, zostanie zastąpiony przez tę wartość parametru **CAPEXPRY** . Zostanie użyty czas utraty ważności określony przez aplikację, który jest mniejszy niż wartość **CAPEXPRY** .

Należy zauważyć, że wartość **CAPEXPRY** jest wyrażona w dziesiątych częściach sekundy, więc jedna minuta ma wartość 600.

Jeśli w ścieżce rozstrzygania używany jest więcej niż jeden obiekt, na przykład gdy komunikat jest umieszczany w kolejce aliasowej lub zdalnej, jako górny limit utraty ważności komunikatu używana jest najniższa ze wszystkich wartości parametru **CAPEXPRY** .

Zmiany wartości parametru **CAPEXPRY** są uwzględniane natychmiast. Wartość utraty ważności jest wartościowana dla każdej operacji umieszczania w kolejce lub temacie i dlatego jest wrażliwa na rozstrzyganie obiektu, które może być różne dla każdej operacji umieszczania.

Należy jednak zauważyć, że zmiana nie ma wpływu na istniejące komunikaty w kolejce przed zmianą w produkcie **CAPEXPRY**(tzn. ich czas utraty ważności pozostaje niezmieniony). Tylko nowe komunikaty, które są umieszczane w kolejce po zmianie w produkcie **CAPEXPRY** , mają nowy czas utraty ważności.

W klastrze, w którym wykonywane jest umieszczenie w kolejce otwartej z opcją MQOO_BIND_NOT_FIXED, do komunikatów można przypisać różne wartości utraty ważności dla każdego umieszczenia, w zależności od wartości **CAPEXPRY** ustawionej dla kolejki transmisji używanej przez kanał, który wysyła komunikat do wybranego docelowego menedżera kolejek.

Należy zauważyć, że umieszczenie w kolejce lub temacie przez aplikację JMS określającą opóźnienie dostarczenia kończy się niepowodzeniem z błędem MQRC_EXPIRY_ERROR, jeśli opóźnienie dostarczenia przekracza rozstrzygnięty czas utraty ważności dla docelowej kolejki lub tematu. Ten błąd może być spowodowany przez ustawienie atrybutu **CAPEXPRY** w kolejce rozstrzygniętej dla miejsca docelowego JMS .

Uwaga: Parametr **CAPEXPRY** nie może być używany w żadnych kolejkach, w których będą przechowywane IBM MQ komunikaty generowane wewnętrznie, takie jak SYSTEM.CLUSTER.* i systemu SYSTEM.PROTECTION.POLICY.QUEUE.

Odsyłacze pokrewne

DEFINE kolejki

Temat DEFINE

Informacje zwrotne (MQLONG) dla deskryptora MQMD

Pole Feedback jest używane z komunikatem typu MQMT_REPORT w celu wskazania rodzaju raportu i ma znaczenie tylko w przypadku tego typu komunikatu.

Pole może zawierać jedną z wartości MQFB_ * lub jedną z wartości MQRC_ *. Kody sprzężenia zwrotnego są pogrupowane w następujący sposób:

MQFB_BRAK
Nie podano informacji zwrotnej.

MQFB_SYSTEM_FIRST
Najniższa wartość informacji zwrotnej generowanej przez system.

MQFB_SYSTEM_LAST
Najwyższa wartość dla informacji zwrotnych wygenerowanych przez system.

Zakres wygenerowanych przez system kodów sprzężenia zwrotnego od MQFB_SYSTEM_FIRST do MQFB_SYSTEM_LAST obejmuje ogólne kody sprzężenia zwrotnego wymienione w tym temacie (MQFB_ *), a także kody przyczyny (MQRC_ *), które mogą wystąpić, gdy komunikat nie może zostać umieszczony w kolejce docelowej.

MQFB_APPL_FIRST
Najniższa wartość dla informacji zwrotnych wygenerowanych przez aplikację.

MQFB_APPL_LAST
Najwyższa wartość dla informacji zwrotnych wygenerowanych przez aplikację.

Aplikacje generujące komunikaty raportów nie mogą używać kodów sprzężenia zwrotnego z zakresu systemu (innych niż MQFB_QUIT), chyba że chcą symulować komunikaty raportu wygenerowane przez menedżera kolejek lub agenta kanału komunikatów.

W wywołaniach MQPUT lub MQPUT1 podana wartość musi być równa MQFB_NONE lub należeć do zakresu systemu lub zakresu aplikacji. Ta opcja jest sprawdzana niezależnie od wartości parametru *MsgType*.

Ogólne kody sprzężenia zwrotnego:

MQFB_COA
Potwierdzenie przybycia do kolejki docelowej (patrz MQRO_COA).

MQFB_COD
Potwierdzenie dostarczenia do aplikacji odbierającej (patrz MQRO_COD).

MQFB_UTR_WAŻN.
Komunikat został odrzucony, ponieważ nie został usunięty z kolejki docelowej przed upływem czasu ważności.

MQFB_PAN
Powiadomienie o działaniu pozytywnym (patrz MQRO_PAN).

MQFB_NAN
Powiadomienie o negatywnym działaniu (patrz MQRO_NAN).

MQFB_QUIT
Zakończ działanie aplikacji.

Może być ona używana przez program do planowania obciążenia do sterowania liczbą uruchomionych instancji aplikacji. Wysłanie komunikatu MQMT_REPORT z tym kodem sprzężenia zwrotnego do instancji aplikacji wskazuje tej instancji, że powinna ona zatrzymać przetwarzanie. Jednak przestrzeganie tej konwencji jest sprawą aplikacji i nie jest wymuszane przez menedżer kolejek.

Kody sprzężenia zwrotnego kanału:

ZAKOŃCZONE_KANAŁU_MQFB_CHANN_XX_ENCODE_CASE_CAPS_LOCK_OFF
Kanał został zakończony normalnie.

MQFB_CHANNEL_FAIL
Kanał został zakończony nieprawidłowo i przechodzi w stan STOPPED.

MQFB_CHANNEL_FAIL_RETRY
Kanał został zakończony nieprawidłowo i przechodzi w stan RETRY.

Kody sprzężenia zwrotnego mostuIMS-bridge

Te kody są używane po odebraniu nieoczekiwanego kodu rozpoznania IMS-OTMA. Kod diagnostyczny lub, jeśli kod diagnostyczny ma wartość 0x1A , kod przyczyny powiązany z tym kodem diagnostycznym jest wskazywany w polu *Feedback*(Opinia).

1. Dla kodów *Feedback* z zakresu od MQFB_IMS_FIRST (300) do MQFB_IMS_LAST (399) odebrano kod rozpoznania inny niż 0x1A . *Kod diagnostyczny* jest podawany przez wyrażenie (*Feedback* - MQFB_IMS_FIRST+1)
2. Dla kodów *Feedback* w zakresie od MQFB_IMS_NACK_1A_REASON_FIRST (600) do MQFB_IMS_NACK_1A_REASON_LAST (855) odebrano kod rozpoznania 0x1A . *Kod przyczyny* powiązany z kodem diagnostycznym jest podawany przez wyrażenie (*Feedback* - MQFB_IMS_NACK_1A_REASON_FIRST)

Znaczenie kodów rozpoznania IMS-OTMA i odpowiadających im kodów przyczyny opisano w podręczniku *Open Transaction Manager Access Guide and Reference*.

Most IMS może wygenerować następujące kody sprzężenia zwrotnego:

MQFB_DATA_LENGTH_ZERO
Długość segmentu w danych aplikacji komunikatu wynosiła zero.

MQFB_DATA_LENGTH_NEGATIVE
Długość segmentu była ujemna w danych aplikacji komunikatu.

MQFB_DATA_LENGTH_TOO_BIG
Długość segmentu była zbyt duża w danych aplikacji komunikatu.

Przepełnienie buforu MQFB_BUFFER_OVERFLOW
Wartość jednego z pól długości spowodowałaby przepełnienie buforu komunikatów przez dane.

MQFB_LENGTH_OFF_BY_ONE
Wartość jednego z pól długości była o 1 bajt za krótka.

MQFB_IIH_BŁĄD
Pole *Format* w strukturze MQMD określa strukturę MQFMT_IMS, ale komunikat nie rozpoczyna się od poprawnej struktury MQIIH.

MQFB_NOT_AUTHORIZED_FOR_IMS (MQFB_NIE_AUTHORITY)
Sprawdzenie poprawności przez most IMS nie powiodło się dla identyfikatora użytkownika zawartego w deskryptorze komunikatu MQMD lub hasła zawartego w polu *Authenticator* w strukturze MQIIH. W rezultacie komunikat nie został przekazany do programu IMS.

BŁĄD MQFB_IMS_ERROR
IMSzwrócił nieoczekiwany błąd. Więcej informacji na temat błędu można znaleźć w dzienniku błędów systemu IBM MQ , w którym znajduje się most IMS .

MQFB_IMS_FIRST
Jeśli kod rozpoznania IMS-OTMA nie jest kodem 0x1A, IMSwygenerowane kody sprzężenia zwrotnego należą do zakresu od MQFB_IMS_FIRST (300) do MQFB_IMS_LAST (399). Sam kod diagnostyczny IMS-OTMA to *Feedback* minus MQFB_IMS_ERROR.

MQFB_IMS_LAST
Najwyższa wartość dla informacji zwrotnej wygenerowanych przez IMS, gdy kod diagnostyczny nie jest równy 0x1A.

MQFB_IMS_NACK_1A_REASON_FIRST
Jeśli kod rozpoznania to 0x1A, wygenerowane przez IMSkody sprzężenia zwrotnego należą do zakresu od MQFB_IMS_NACK_1A_REASON_FIRST (600) do MQFB_IMS_NACK_1A_REASON_LAST (855).

MQFB_IMS_NACK_1A_REASON_LAST
Najwyższa wartość dla informacji zwrotnych wygenerowanych przez IMS, gdy kod diagnostyczny ma wartość 0x1A

CICS-bridge feedback codes: CICS bridgemoże wygenerować następujące kody sprzężenia zwrotnego:

MQFB_CICS_APPL_ABENDED
Aplikacja określona w komunikacie została zakończona nieprawidłowo. Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_APPL_NOT_STARTED (nie uruchomiono)
Operacja EXEC CICS LINK dla aplikacji określonej w komunikacie nie powiodła się. Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_BRIDGE_FAILURE
Program CICS bridge został zakończony nieprawidłowo bez zakończenia normalnego przetwarzania błędów.

BŁĄD MQFB_CICS_CCSID_ERROR
Niepoprawny identyfikator zestawu znaków.

BŁĄD MQFB_CICS_CIH_ERROR
Brak lub niepoprawna struktura nagłówka informacji CICS .

BŁĄD MQFB_CICS_COMMAREA_ERROR
Niepoprawna długość obszaru CICS COMMAREA.

BŁĄD MQFB_CICS_CORREL_ID_ERROR
Niepoprawny identyfikator korelacji.

BŁĄD MQFB_CICS_DLQ_ERROR
Zadanie CICS bridge nie mogło skopiować odpowiedzi na to żądanie do kolejki niedostarczonych komunikatów. Żądanie zostało wycofane.

BŁĄD WYWOŁANIA MQFB_CICS_ENCODING_ERROR
Niepoprawne kodowanie.

MQFB_CICS_INTERNAL_ERROR (Błąd interfejsu CICS)
Program CICS bridge napotkał nieoczekiwany błąd.

Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_NOT_AUTHORIZED
Identyfikator użytkownika nie jest autoryzowany lub hasło jest niepoprawne.

Ten kod sprzężenia zwrotnego występuje tylko w polu *Reason* struktury MQDLH.

MQFB_CICS_UOW_BACKED_OUT
Jednostka pracy została wycofana z jednego z następujących powodów:

- Wykryto niepowodzenie podczas przetwarzania innego żądania w tej samej jednostce pracy.
- W trakcie trwania jednostki pracy wystąpiło nieprawidłowe zakończenie CICS .

BŁĄD MQFB_CICS_UOW_ERROR
Niepoprawne pole sterujące jednostki pracy *UOWControl* .

Kody sprzężenia zwrotnego komunikatu trasy śledzenia:

MQFB_ACTIVITY
Używany w połączeniu z formatem MQFMT_EMBEDDED_PCF w celu umożliwienia opcji danych użytkownika po raportach działań.

DZIAŁANIA MQFB_MAKSIMUM
Zwracany, gdy komunikat trasy śledzenia jest odrzucany, ponieważ liczba działań, w które uczestniczył komunikat, przekracza limit maksymalnej liczby działań.

MQFB_NOT_FORWARDED
Zwracany, gdy komunikat trasy śledzenia zostanie odrzucony, ponieważ ma zostać wysłany do zdalnego menedżera kolejek, który nie obsługuje komunikatów trasy śledzenia.

MQFB_NIEDELIVERED
Zwracany, gdy komunikat trasy śledzenia zostanie odrzucony, ponieważ ma zostać umieszczony w kolejce lokalnej.

MQFB_UNSUPPORTED_FORWARDING
Zwracany, gdy komunikat trasy śledzenia zostanie odrzucony, ponieważ wartość parametru przekazywania jest nierozpoznana i znajduje się w odrzuconej masce bitowej.

MQFB_UNSUPPORTED_DELIVERY
Zwracany, gdy komunikat trasy śledzenia jest odrzucany, ponieważ wartość w parametrze dostarczania jest nierozpoznana i znajduje się w odrzuconej masce bitowej.

IBM MQ kody przyczyny: w przypadku komunikatów raportów o wyjątkach *Feedback* zawiera kod przyczyny IBM MQ . Możliwe są następujące kody przyczyny:

MQRC_PUT_INHIBITED
(2051, X'803 ') Wywołania umieszczania zablokowane dla kolejki.

MQRC_Q_FULL
(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') Brak miejsca na dysku dla kolejki.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') Kolejka nie obsługuje komunikatów trwałych.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Długość komunikatu przekracza maksimum dla kolejki.

Pełną listę kodów przyczyny można znaleźć pod adresem:

- W przypadku systemu IBM MQ for z/OSnależy zapoznać się z sekcją Kody zakończenia i kody przyczyny interfejsu API.
- W przypadku wszystkich innych platform należy zapoznać się z sekcją Kody zakończenia i kody przyczyny interfejsu API.

.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQFB_NONE.

Kodowanie (MQLONG) dla deskryptora MQMD

Określa kodowanie liczbowe danych liczbowych w komunikacie. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQMD. Kodowanie liczbowe definiuje reprezentację używaną dla binarnych liczb całkowitych, upakowanych liczb całkowitych i liczb zmiennopozycyjnych.

W systemie z/OSbinarna część całkowita pola `Encoding` jest również używana do określania kodowania danych znakowych w treści komunikatu w postaci liczby całkowitej, gdy odpowiedni identyfikator zestawu znaków wskazuje, że reprezentacja zestawu znaków jest zależna od kodowania używanego dla binarnych liczb całkowitych. Dotyczy to tylko niektórych zestawów znaków wielobajtowych (na przykład zestawów znaków UTF-16).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Zdefiniowana jest następująca wartość specjalna:

RODZIMA MQENC
Kodowanie jest domyślne dla języka programowania i komputera, na którym działa aplikacja.

Uwaga: Wartość tej stałej zależy od języka programowania i środowiska. Z tego powodu aplikacje muszą być kompilowane przy użyciu plików nagłówka, makra, COPY lub INCLUDE odpowiednich dla środowiska, w którym aplikacja będzie uruchamiana.

Aplikacje, które umieszczają komunikaty, zwykle określają parametr MQENC_NATIVE. Aplikacje, które pobierają komunikaty, muszą porównać to pole z wartością MQENC_NATIVE; jeśli wartości różnią się, może być konieczne przekształcenie danych liczbowych w komunikacie przez aplikację. Użyj opcji MQGMO_CONVERT, aby zażądać od menedżera kolejek przekształcenia komunikatu w ramach przetwarzania wywołania MQGET. Szczegółowe informacje na temat tworzenia pola `Encoding` zawiera sekcja “Kodowania maszynowe” na stronie 930 .

Jeśli w wywołaniu MQGET zostanie podana opcja MQGMO_CONVERT, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest kodowaniem, na które w razie potrzeby mają zostać przekształcone dane komunikatu. Jeśli konwersja powiedzie się lub nie będzie potrzebna, wartość pozostanie niezmieniona. Jeśli konwersja nie powiedzie się, wartość po wywołaniu MQGET reprezentuje kodowanie nieprzekształconego komunikatu, który jest zwracany do aplikacji.

W innych przypadkach jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQENC_NATIVE.

CodedCharSetId (MQLONG) dla MQMD

To pole określa identyfikator zestawu znaków danych znakowych w treści komunikatu.

Uwaga: Dane znakowe w strukturze MQMD i innych strukturach danych produktu MQ , które są parametrami wywołań, muszą znajdować się w zestawie znaków menedżera kolejek. Jest on definiowany przez atrybut **CodedCharSetId** menedżera kolejek. Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty menedżera kolejek” na stronie 824 .

Jeśli to pole jest ustawione na wartość MQCCSI_Q_MGR podczas wywoływania komendy MQGET z opcją MQGMO_CONVERT w opcjach, zachowanie jest inne w przypadku aplikacji klienckich i aplikacji serwera. W przypadku aplikacji serwera stroną kodową używaną do konwersji znaków jest `CodedCharSetId` menedżera kolejek. W przypadku aplikacji klienckich stroną kodową używaną do konwersji znaków jest bieżąca strona kodowa ustawień narodowych.

W przypadku aplikacji klienckich pole MQCCSI_Q_MGR jest wypełniane na podstawie ustawień narodowych klienta, a nie w menedżerze kolejek. Wyjątkiem od tej reguły jest umieszczanie komunikatu w kolejce mostu IMS . W polu *`CodedCharSetId`* deskryptora MQMD zwracany jest identyfikator CCSID menedżera kolejek.

Nie można używać następującej wartości specjalnej:

APPL MQCCSI_PL
Powoduje to podanie niepoprawnej wartości w polu CodedCharSetId deskryptora MQMD i powoduje zwrócenie kodu powrotu MQRC_SOURCE_CCSID_ERROR (lub MQRC_FORMAT_ERROR dla systemu z/OS). po odebraniu komunikatu przy użyciu wywołania MQGET z opcją MQGMO_CONVERT.

Można użyć następujących wartości specjalnych:

MQCCSI_Q_MGR (menedżer kolejek MQ)
Dane znakowe w komunikacie znajdują się w zestawie znaków menedżera kolejek.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia tę wartość w deskryptorze MQMD, który jest wysyłany z komunikatem, na prawdziwy identyfikator zestawu znaków menedżera kolejek. W rezultacie wartość MQCCSI_Q_MGR nigdy nie jest zwracana przez wywołanie MQGET.

MQCCSI_DEFAULT
Wartość CodedCharSetId danych w polu *String* jest definiowana przez pole CodedCharSetId w strukturze nagłówka, która poprzedza strukturę MQCFH, lub przez pole CodedCharSetId w strukturze MQMD, jeśli MQCFH jest na początku komunikatu.

MQCCSI_INHERIT
Dane znakowe w komunikacie znajdują się w tym samym zestawie znaków, co ta struktura. Jest to zestaw znaków menedżera kolejek. (Tylko dla MQMD, MQCCSI_INHERIT ma takie samo znaczenie jak MQCCSI_Q_MGR).

Menedżer kolejek zmienia tę wartość w deskryptorze MQMD wysyłanym z komunikatem na rzeczywisty identyfikator zestawu znaków MQMD. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie należy używać pola MQCCSI_INHERIT, jeśli wartością pola PutApplType w deskryptorze MQMD jest MQAT_BROKER.

MQCCSI_EMBEDDED
Dane znakowe w komunikacie znajdują się w zestawie znaków z identyfikatorem zawartym w samych danych komunikatu. W danych komunikatu może być osadzona dowolna liczba identyfikatorów zestawów znaków, które mają zastosowanie do różnych części danych. Ta wartość musi być używana dla komunikatów PCF (w formacie MQFMT_ADMIN, MQFMT_EVENT lub MQFMT_PCF), które zawierają dane w mieszance zestawów znaków. Każda struktura MQCFST, MQCFSL i MQCFSF zawarta w komunikacie PCF musi mieć określony jawny identyfikator zestawu znaków, a nie MQCCSI_DEFAULT.

Jeśli komunikat w formacie MQFMT_EMBEDDED_PCF ma zawierać dane w mieszance zestawów znaków, nie należy używać parametru MQCCSI_EMBEDDED. Zamiast tego należy ustawić opcję MQEPH_CCSID_EMBEDDED w polu Flags w strukturze MQEPH. Jest to równoważne ustawieniu parametru MQCCSI_EMBEDDED w poprzedniej strukturze. Każda struktura MQCFST, MQCFSL i MQCFSF zawarta w komunikacie PCF musi mieć określony jawny identyfikator zestawu znaków, a nie MQCCSI_DEFAULT. Więcej informacji na temat struktury MQEPH zawiera sekcja “MQEPH-wbudowany nagłówek PCF” na stronie 371.

Tę wartość należy podać tylko w wywołaniach MQPUT i MQPUT1 . Jeśli jest ona określona w wywołaniu MQGET, uniemożliwia konwersję komunikatu.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia wartości MQCCSI_Q_MGR i MQCCSI_INHERIT w strukturze MQMD wysyłanej z komunikatem w sposób opisany powyżej, ale nie zmienia deskryptora MQMD określonego w wywołaniu MQPUT lub MQPUT1 . Dla podanej wartości nie jest wykonywane żadne inne sprawdzenie.

Aplikacje, które pobierają komunikaty, muszą porównać to pole z wartością oczekiwaną przez aplikację. Jeśli wartości różnią się, aplikacja może potrzebować konwersji danych znakowych w komunikacie.

W systemie z/OSpole Encoding deskryptora MQMD służy do określania kodowania danych znakowych w treści komunikatu na podstawie liczby całkowitej, gdy pole CodedCharSetId deskryptora MQMD wskazuje, że reprezentacja zestawu znaków jest zależna od kodowania używanego dla binarnych liczb całkowitych. W systemie Wiele platformprzyjmuje się, że kolejność bajtów danych znakowych jest taka

sama jak kodowanie rodzimej liczby całkowitej dla platformy, na której działa menedżer kolejek. Dotyczy to tylko niektórych zestawów znaków wielobajtowych (na przykład zestawów znaków UTF-16).

Jeśli w wywołaniu MQGET zostanie podana opcja MQGMO_CONVERT, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest identyfikatorem kodowanego zestawu znaków, na który w razie potrzeby mają zostać przekształcone dane komunikatu. Jeśli konwersja jest pomyślna lub niepotrzebna, wartość pozostaje niezmieniona (z wyjątkiem tego, że wartość MQCCSI_Q_MGR lub MQCCSI_INHERIT jest przekształcana w wartość rzeczywistą). Jeśli konwersja nie powiedzie się, wartość po wywołaniu MQGET reprezentuje identyfikator kodowanego zestawu znaków nieprzekształconego komunikatu, który jest zwracany do aplikacji.

W przeciwnym razie jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQCCSI_Q_MGR.

Format (MQCHAR8) dla deskryptora MQMD

Jest to nazwa używana przez nadawcę komunikatu do wskazania odbiorcy rodzaju danych w komunikacie. Dla nazwy można określić dowolne znaki znajdujące się w zestawie znaków menedżera kolejek, ale należy ograniczyć nazwę do następujących elementów:

- Wielkie litery od A do Z
- Cyfry od 0 do 9

Jeśli używane są inne znaki, translacja nazwy między zestawami znaków wysyłającego i odbierającego menedżera kolejek może być niemożliwa.

Nazwę należy dopełnić spacjami do długości pola lub użyć znaku o kodzie zero, aby zakończyć nazwę przed końcem pola. Znak o kodzie zero i wszystkie kolejne znaki są traktowane jako odstępy. Nie należy podawać nazwy z odstępami wiodącymi lub osadzonymi. W przypadku wywołania MQGET menedżer kolejek zwraca nazwę dopełnianą spacjami do długości pola.

Menedżer kolejek nie sprawdza, czy nazwa jest zgodna z zaleceniami opisanymi powyżej.

Nazwy rozpoczynające się od liter MQ pisane wielkimi, małymi i mieszanymi literami mają znaczenie zdefiniowane przez menedżer kolejek. W przypadku własnych formatów nie należy używać nazw zaczynających się od tych liter. Wbudowane formaty menedżera kolejek są następujące:

MQFMT_BRAK
Rodzaj danych jest niezdefiniowany: danych nie można przekształcić, gdy komunikat jest pobierany z kolejki przy użyciu opcji MQGMO_CONVERT.

Jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT, a zestaw znaków lub kodowanie danych w komunikacie różni się od określonego w parametrze **MsgDesc** , komunikat jest zwracany z następującymi kodami zakończenia i przyczyny (przy założeniu braku innych błędów):

- Kod zakończenia MQCC_WARNING i kod przyczyny MQRC_FORMAT_ERROR, jeśli dane MQFMT_NONE znajdują się na początku komunikatu.
- Kod zakończenia MQCC_OK i kod przyczyny MQRC_NONE, jeśli dane MQFMT_NONE znajdują się na końcu komunikatu (czyli przed jedną lub większą liczbą struktur nagłówka MQ). W tym przypadku struktury nagłówków MQ są przekształcane w żądany zestaw znaków i kodowanie.

W języku programowania C jest również zdefiniowana stała MQFMT_NONE_ARRAY. Ma ona taką samą wartość jak MQFMT_NONE, ale jest tablicą znaków zamiast łańcucha.

MQFMT_ADMIN,
Komunikat jest żądaniem serwera komend lub komunikatem odpowiedzi w formacie komend programowalnych (PCF). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT. Więcej informacji na temat korzystania z komunikatów w formacie komend programowalnych zawiera sekcja Korzystanie z formatów komend programowalnych .

W języku programowania C zdefiniowana jest również stała MQFMT_ADMIN_ARRAY. Ma ona taką samą wartość jak MQFMT_ADMIN, ale jest tablicą znaków zamiast łańcucha.

MQFMT_CICS
Dane komunikatu rozpoczynają się od nagłówka informacji CICS MQCIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole `Format` w strukturze MQCIH.

z/OS W systemie z/OSnależy określić opcję MQGMO_CONVERT w wywołaniu MQGET, aby przekształcić komunikaty w formacie MQFMT_CICS.

W języku programowania C jest również zdefiniowana stała MQFMT_CICS_ARRAY. Ma ona taką samą wartość jak MQFMT_CICS, ale jest tablicą znaków zamiast łańcucha.

MQFMT_COMMAND_1
Komunikat jest komunikatem odpowiedzi serwera komend MQSC zawierającym liczbę obiektów, kod zakończenia i kod przyczyny. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

W języku programowania C zdefiniowana jest również stała MQFMT_COMMAND_1_ARRAY , która ma taką samą wartość jak MQFMT_COMMAND_1, ale jest tablicą znaków zamiast łańcucha.

MQFMT_COMMAND_2
Komunikat jest komunikatem odpowiedzi serwera komend MQSC zawierającym informacje o żądanych obiektach. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania C zdefiniowana jest również stała MQFMT_COMMAND_2_ARRAY ; ma ona taką samą wartość jak MQFMT_COMMAND_2, ale jest tablicą znaków zamiast łańcucha.

MQFMT_DEAD_LETTER_HEADER
Dane komunikatu rozpoczynają się od nagłówka MQDLH niedostarczonego komunikatu. Dane z oryginalnego komunikatu są bezpośrednio zgodne ze strukturą MQDLH. Nazwa formatu oryginalnych danych komunikatu jest podana w polu `Format` w strukturze MQDLH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQDLH-nagłówek niedostarczonego komunikatu” na stronie 359 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Raporty COA i COD nie są generowane dla komunikatów z atrybutem `Format` o wartości MQFMT_DEAD_LETTER_HEADER.

Dla języka programowania C zdefiniowana jest również stała MQFMT_DEAD_LETTER_HEADER_ARRAY. Ma ona taką samą wartość jak MQFMT_DEAD_LETTER_HEADER, ale jest tablicą znaków, a nie łańcuchem.

Usługa MQFMT_DIST_HEADER
Dane komunikatu rozpoczynają się od nagłówka listy dystrybucyjnej MQDH. Obejmuje to tablice rekordów MQOR i MQPMR. Po nagłówku listy dystrybucyjnej mogą następować dodatkowe dane. Format dodatkowych danych (jeśli istnieją) jest podany w polu *`Format`* w strukturze MQDH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQDH-nagłówek dystrybucji” na stronie 353 . Komunikaty w formacie MQFMT_DIST_HEADER można przekształcić, jeśli w wywołaniu MQGET podano opcję MQGMO_CONVERT.

Ten format jest obsługiwany w następujących środowiskach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

Dla języka programowania C zdefiniowana jest również stała MQFMT_DIST_HEADER_ARRAY. Ma ona taką samą wartość jak MQFMT_DIST_HEADER, ale jest tablicą znaków, a nie łańcuchem.

MQFMT_EMBEDDED_PCF
Format komunikatu trasy śledzenia, pod warunkiem, że wartość komendy PCF jest ustawiona na MQCMD_TRACE_ROUTE. Użycie tego formatu umożliwia wysyłanie danych użytkownika wraz z komunikatem trasy śledzenia, pod warunkiem, że ich aplikacje mogą obsłużyć wcześniejsze parametry PCF.

Nagłówek PCF musi być pierwszym nagłówkiem, w przeciwnym razie komunikat nie będzie traktowany jako komunikat trasy śledzenia. Oznacza to, że komunikat nie może znajdować się w grupie i że komunikaty trasy śledzenia nie mogą być segmentowane. Jeśli komunikat trasy śledzenia jest wysyłany w grupie, komunikat jest odrzucany z kodem przyczyny MQRC_MSG_NOT_ALLOWED_IN_GROUP.

Należy zauważyć, że parametr MQFMT_ADMIN może być również używany dla formatu komunikatu trasy śledzenia, ale w tym przypadku nie można wysłać danych użytkownika razem z komunikatem trasy śledzenia.

MQFMT_EVENT, ZDARZENIE
Komunikat jest komunikatem zdarzenia MQ , który zgłasza wystąpienie zdarzenia. Komunikaty zdarzeń mają taką samą strukturę jak komendy programowalne. Więcej informacji na temat tej struktury zawiera sekcja Komunikaty komend PCF , a informacje na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń .

Komunikaty zdarzeń w wersji Version-1 mogą być przekształcane we wszystkich środowiskach, jeśli w wywołaniu MQGET podano opcję MQGMO_CONVERT. Komunikaty zdarzeń Version-2 mogą być przekształcane tylko w systemie z/OS.

W języku programowania C zdefiniowana jest również stała MQFMT_EVENT_ARRAY. Ma ona taką samą wartość jak MQFMT_EVENT, ale jest tablicą znaków, a nie łańcuchem.

MQFMT_IMS,
Dane komunikatu rozpoczynają się od nagłówka informacji IMS MQIIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole `Format` w strukturze MQIIH.

Szczegółowe informacje na temat sposobu obsługi struktury MQIIH podczas używania komendy MQGET z opcją MQGMO_CONVERT zawiera sekcja “Format (MQCHAR8) dla MQIIH” na stronie 420 i “ReplyToFormat (MQCHAR8) dla MQIIH” na stronie 421.

W języku programowania C zdefiniowana jest również stała MQFMT_IMS_ARRAY. Ma ona taką samą wartość jak MQFMT_IMS, ale jest tablicą znaków, a nie łańcuchem.

MQFMT_IMS_VAR_STRING (łańcuch zmiennych)
Komunikat jest łańcuchem zmiennej IMS , który jest łańcuchem w postaci `ll`*`zz`*`ccc`, gdzie:

`ll`
to pole o długości 2 bajtów określające całkowitą długość elementu łańcucha zmiennej IMS . Ta długość jest równa długości `ll` (2 bajty) plus długości *`zz`* (2 bajty) plus długości samego łańcucha znaków. `ll` jest 2-bajtową binarną liczbą całkowitą w kodowaniu określonym przez pole `Encoding` .

`zz`
to pole dwubajtowe zawierające flagi istotne dla IMS. *`zz`* jest łańcuchem bajtowym składającym się z dwóch pól MQBYTE i jest przesyłany bez zmiany nadawcy na odbiorcę (czyli *`zz`* nie podlega żadnej konwersji).

`ccc`
jest łańcuchem znaków o zmiennej długości, zawierającym znaki `ll-4` . *`ccc`* znajduje się w zestawie znaków określonym w polu `CodedCharSetId` .

W systemie z/OSdane komunikatu mogą składać się z sekwencji łańcuchów zmiennych IMS , przy czym każdy łańcuch ma postać `llzzccc`. Między kolejnymi łańcuchami zmiennych IMS nie mogą być pomijane żadne bajty. Oznacza to, że jeśli pierwszy łańcuch ma nieparzystą długość, drugi łańcuch nie będzie wyrównany, to znaczy nie będzie rozpoczynał się od granicy, która jest wielokrotnością dwóch. Należy zachować ostrożność podczas konstruowania takich łańcuchów na maszynach, które wymagają wyrównania podstawowych typów danych.

Użyj opcji MQGMO_CONVERT w wywołaniu MQGET, aby przekształcić komunikaty w formacie MQFMT_IMS_VAR_STRING.

Dla języka programowania C zdefiniowana jest również stała MQFMT_IMS_VAR_STRING_ARRAY. Ma ona taką samą wartość jak MQFMT_IMS_VAR_STRING, ale jest tablicą znaków zamiast łańcucha.

MQFMT_MD_EXTENSION
Dane komunikatu rozpoczynają się od rozszerzenia deskryptora komunikatu MQMDE i opcjonalnie następują po nich inne dane (zazwyczaj dane komunikatu aplikacji). Nazwa formatu, zestaw znaków i kodowanie danych po MQMDE są podane w polach `Format`, `CodedCharSetId`i `Encoding` w MQMDE. Szczegółowe informacje na temat tej struktury zawiera sekcja "MQMDE-rozszerzenie deskryptora komunikatu" na stronie 485 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania C jest również zdefiniowana stała MQFMT_MD_EXTENSION_ARRAY. Ma ona taką samą wartość jak MQFMT_MD_EXTENSION, ale jest tablicą znaków zamiast łańcucha.

MQFMT_PCF,
Komunikat jest komunikatem zdefiniowanym przez użytkownika, który jest zgodny ze strukturą komunikatu PCF (programmable command format). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT. Więcej informacji na temat korzystania z komunikatów w formacie komend programowalnych zawiera sekcja Korzystanie z formatów komend programowalnych .

Dla języka programowania C zdefiniowana jest również stała MQFMT_PCF_ARRAY. Ma ona taką samą wartość jak MQFMT_PCF, ale jest tablicą znaków, a nie łańcuchem.

MQFMT_REF_MSG_HEADER
Dane komunikatu rozpoczynają się od nagłówka komunikatu odniesienia MQRMH, po którym opcjonalnie następują inne dane. Nazwa formatu, zestaw znaków i kodowanie danych są nadawane przez pola `Format`, `CodedCharSetId`i `Encoding` w MQRMH. Szczegółowe informacje na temat tej struktury zawiera sekcja "MQRMH-nagłówek komunikatu referencyjnego" na stronie 565 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Ten format jest obsługiwany w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

W języku programowania C jest również zdefiniowana stała MQFMT_REF_MSG_HEADER_ARRAY. Ma ona taką samą wartość jak MQFMT_REF_MSG_HEADER, ale jest tablicą znaków zamiast łańcucha.

ZMQFMT_RF_HEADER
Dane komunikatu rozpoczynają się od nagłówka MQRFH reguł i formatowania, po którym opcjonalnie następują inne dane. Nazwa formatu, zestaw znaków i kodowanie danych (jeśli istnieją) są nadawane przez pola `Format`, `CodedCharSetId`i `Encoding` w MQRFH. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

W języku programowania C jest również zdefiniowana stała MQFMT_RF_HEADER_ARRAY. Ma ona taką samą wartość jak MQFMT_RF_HEADER, ale jest tablicą znaków, a nie łańcuchem.

MQFMT_RF_HEADER_2
Dane komunikatu rozpoczynają się od reguł version-2 i nagłówka formatowania MQRFH2, po którym opcjonalnie następują inne dane. Nazwa formatu, zestaw znaków i kodowanie danych opcjonalnych (jeśli istnieją) są nadawane przez pola `Format`, `CodedCharSetId`i `Encoding` w nagłówku MQRFH2.

Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania C zdefiniowana jest również stała MQFMT_RF_HEADER_2_ARRAY . Ma ona taką samą wartość jak zmienna MQFMT_RF_HEADER_2, ale jest tablicą znaków zamiast łańcucha.

MQFMT_STRING
Dane komunikatu aplikacji mogą być łańcuchem SBCS (zestaw znaków jednobajtowych) lub łańcuchem DBCS (zestaw znaków dwubajtowych). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

Dla języka programowania C zdefiniowana jest również stała MQFMT_STRING_ARRAY. Ma ona taką samą wartość jak MQFMT_STRING, ale jest tablicą znaków zamiast łańcucha.

MQFMT_TRIGGER,
Komunikat jest komunikatem wyzwalacza opisanym przez strukturę MQTM. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQTM-komunikat wyzwalacza” na stronie 618 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT.

W języku programowania C zdefiniowana jest również stała MQFMT_TRIGGER_ARRAY. Ma ona taką samą wartość jak MQFMT_TRIGGER, ale jest tablicą znaków, a nie łańcuchem.

MQFMT_WORK_INFO_HEADER,
Dane komunikatu rozpoczynają się od nagłówka informacji o pracy MQWIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest nadawana przez pole `Format` w strukturze MQWIH.

z/OS W systemie z/OSnależy określić opcję MQGMO_CONVERT w wywołaniu MQGET, aby przekształcić dane użytkownika w komunikatach, które mają format MQFMT_WORK_INFO_HEADER. Jednak sama struktura MQWIH jest zawsze zwracana w zestawie znaków i kodowaniu menedżera kolejek (struktura MQWIH jest przekształcana bez względu na to, czy podano opcję MQGMO_CONVERT).

Dla języka programowania C zdefiniowana jest również stała MQFMT_WORK_INFO_HEADER_ARRAY. Ma ona taką samą wartość jak MQFMT_WORK_INFO_HEADER, ale jest tablicą znaków zamiast łańcucha.

MQFMT_XMIT_Q_HEADER
Dane komunikatu rozpoczynają się od nagłówka kolejki transmisji MQXQH. Dane z oryginalnego komunikatu są bezpośrednio zgodne ze strukturą MQXQH. Nazwa formatu oryginalnych danych komunikatu jest nadawana przez pole `Format` w strukturze MQMD, która jest częścią nagłówka kolejki transmisji MQXQH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQXQH-nagłówek kolejki transmisji” na stronie 637 .

Raporty COA i COD nie są generowane dla komunikatów, które mają `Format` MQFMT_XMIT_Q_HEADER.

Dla języka programowania C zdefiniowana jest również stała MQFMT_XMIT_Q_HEADER_ARRAY; ma ona taką samą wartość jak MQFMT_XMIT_Q_HEADER, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Priorytet (MQLONG) dla MQMD

W przypadku wywołań MQPUT i MQPUT1 wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następującej wartości specjalnej:

MQPRI_PRIORITY_AS_Q_DEF

- Jeśli kolejka jest kolejką klastra, priorytet komunikatu jest pobierany z atrybutu **DefPriority** zdefiniowanego w *docelowym* menedżerze kolejek, który jest właścicielem konkretnej instancji kolejki, w której został umieszczony komunikat.

Jeśli istnieje wiele instancji kolejki klastra różniących się tym atrybutem, pobierana jest wartość jednego z nich, ale nie można przewidzieć, która z tych wartości zostanie użyta. Dlatego też ten atrybut należy ustawić na tę samą wartość we wszystkich instancjach. Jeśli to nie jest przyczyna problemu, do dzienników menedżera kolejek wysyłany jest komunikat o błędzie AMQ9407. Należy również zapoznać się z sekcją Jak atrybuty obiektów docelowych są rozstrzygane w przypadku kolejek aliasowych, kolejek zdalnych i kolejek klastra?

Wartość *DefPriority* jest kopiowana do pola *Priority* , gdy komunikat jest umieszczany w kolejce docelowej. Jeśli parametr *DefPriority* zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone w kolejce.

- Jeśli kolejka nie jest kolejką klastra, priorytet komunikatu jest pobierany z atrybutu **DefPriority** zdefiniowanego w *lokalnym* menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

  Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, priorytet domyślny jest pobierany z wartości tego atrybutu w *pierwszej* definicji w ścieżce. Może to być:

  – Kolejka aliasowa
  – Kolejka lokalna
  – Lokalna definicja kolejki zdalnej
  – Alias menedżera kolejek
  – Kolejka transmisji (na przykład *DefXmitQName*)

  Wartość *DefPriority* jest kopiowana do pola *Priority* podczas umieszczania komunikatu. Jeśli plik *DefPriority* zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone.

Wartość zwracana przez wywołanie MQGET jest zawsze większa lub równa zero; wartość MQPRI_PRIORITY_AS_Q_DEF nie jest nigdy zwracana.

Jeśli komunikat jest umieszczany z priorytetem większym niż maksimum obsługiwane przez lokalny menedżer kolejek (to maksimum jest nadawane przez atrybut menedżera kolejek **MaxPriority**), komunikat jest akceptowany przez menedżer kolejek, ale umieszczany w kolejce z maksymalnym priorytetem menedżera kolejek. Wywołanie MQPUT lub MQPUT1 kończy działanie z wartością MQCC_WARNING i kodem przyczyny MQRC_PRIORITY_PRZEKROCZENIA maksimum. Jednak pole *Priority* zachowuje wartość określoną przez aplikację, która umieściła komunikat.

W systemie z/OS, jeśli komunikat o numerze MsgSeq1 zostanie umieszczony w kolejce z sekwencją dostarczania komunikatów MQMDS_PRIORITY i typem indeksu MQIT_GROUP_ID, kolejka może traktować komunikat z innym priorytetem. Jeśli komunikat został umieszczony w kolejce z priorytetem 0 lub 1, jest on przetwarzany tak, jakby miał priorytet 2. Wynika to z faktu, że kolejność komunikatów umieszczanych w kolejce tego typu jest zoptymalizowana w celu umożliwienia wydajnych testów kompletności grup. Więcej informacji na temat sekwencji dostarczania komunikatów MQMDS_PRIORITY i typu indeksu MQIT_GROUP_ID zawiera sekcja Atrybut sekwencjiMsgDelivery.

Podczas odpowiadania na komunikat aplikacje muszą używać priorytetu komunikatu żądania dla komunikatu odpowiedzi. W innych sytuacjach określenie wartości MQPRI_PRIORITY_AS_Q_DEF umożliwia strojenie priorytetu bez zmiany aplikacji.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQPRI_PRIORITY_AS_Q_DEF.

Trwałość (MQLONG) dla MQMD

Wskazuje, czy komunikat przetrwa awarie systemu i restarty menedżera kolejek. W przypadku wywołań MQPUT i MQPUT1 wartością musi być jedna z następujących wartości:

MQPER_PERSISTENT
Komunikat przetrwa awarie systemu i restarty menedżera kolejek. Po umieszczeniu komunikatu i zatwierdzeniu jednostki pracy, w której został on umieszczony (jeśli komunikat jest umieszczany jako część jednostki pracy), komunikat jest zachowywany w pamięci dyskowej. Pozostaje tam do momentu

usunięcia komunikatu z kolejki i zatwierdzenia jednostki pracy, w której został on otrzymany (jeśli komunikat jest wczytywany jako część jednostki pracy).

Gdy komunikat trwały jest wysyłany do kolejki zdalnej, mechanizm przechowywania i przekazywania przechowuje komunikat w każdym menedżerze kolejek wzdłuż trasy do miejsca docelowego, dopóki wiadomo, że komunikat dotarł do następnego menedżera kolejek.

Nie można umieścić trwałych komunikatów w:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane, które są odwzorowywane na obiekt CFSTRUCT w CFLEVEL (2) lub poniżej, lub w których obiekt CFSTRUCT jest zdefiniowany jako RECOVER (NO).

Komunikaty trwałe mogą być umieszczane w trwałych kolejkach dynamicznych oraz w predefiniowanych kolejkach.

MQPER_NOT_PERSISTENT
Komunikat zwykle nie przetrwa awarii systemu lub restartów menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartowania menedżera kolejek w pamięci dyskowej zostanie znaleziona nienaruszona kopia komunikatu.

W przypadku kolejek NPMCLASS (HIGH) nietrwałe komunikaty są zachowywane podczas normalnego zamykania i restartowania menedżera kolejek.

W przypadku kolejek współużytkowanych komunikaty nietrwałe są zachowywane po restarcie menedżera kolejek w grupie współużytkowania kolejek, ale nie są zachowywane niepowodzenia narzędzia CF używanego do przechowywania komunikatów w kolejkach współużytkowanych.

MQPER_PERSISTENCE_AS_Q_DEF

- Jeśli kolejka jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu **DefPersistence** zdefiniowanego w *docelowym* menedżerze kolejek, który jest właścicielem konkretnej instancji kolejki, w której został umieszczony komunikat.

  Jeśli istnieje wiele instancji kolejki klastra różniących się tym atrybutem, pobierana jest wartość jednego z nich, ale nie można przewidzieć, która z tych wartości zostanie użyta. Dlatego też ten atrybut należy ustawić na tę samą wartość we wszystkich instancjach. Jeśli to nie jest przyczyna problemu, do dzienników menedżera kolejek wysyłany jest komunikat o błędzie AMQ9407. Należy również zapoznać się z sekcją Jak atrybuty obiektów docelowych są rozstrzygane w przypadku kolejek aliasowych, kolejek zdalnych i kolejek klastra?

  Wartość *DefPersistence* jest kopiowana do pola *Persistence* , gdy komunikat jest umieszczany w kolejce docelowej. Jeśli parametr *DefPersistence* zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone w kolejce.
- Jeśli kolejka nie jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu **DefPersistence** zdefiniowanego w *lokalnym* menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

  Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w *pierwszej* definicji w ścieżce. Może to być:

  – Kolejka aliasowa
  – Kolejka lokalna
  – Lokalna definicja kolejki zdalnej
  – Alias menedżera kolejek
  – Kolejka transmisji (na przykład *DefXmitQName*)

  Wartość *DefPersistence* jest kopiowana do pola *Persistence* podczas umieszczania komunikatu. Jeśli plik *DefPersistence* zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone.

W tej samej kolejce mogą istnieć zarówno komunikaty trwałe, jak i nietrwałe.

Podczas odpowiadania na komunikat aplikacje muszą używać trwałości komunikatu żądania dla komunikatu odpowiedzi.

W przypadku wywołania MQGET zwracana jest wartość MQPER_PERSISTENT lub MQPER_NOT_PERSISTENT.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MQPER_PERSISTENCE_AS_Q_DEF.

MsgId (MQBYTE24) w strukturze MQMD

Jest to łańcuch bajtowy używany do odróżniania jednego komunikatu od drugiego. Ogólnie rzecz biorąc, żaden z dwóch komunikatów nie powinien mieć tego samego identyfikatora komunikatu, chociaż nie jest to niedozwolone przez menedżer kolejek. Identyfikator komunikatu jest trwałą właściwością komunikatu i jest zachowywany po restarcie menedżera kolejek. Ponieważ identyfikator komunikatu jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator komunikatu nie jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do innego.

W przypadku wywołań MQPUT i MQPUT1 , jeśli aplikacja określa opcję MQMI_NONE lub MQPMO_NEW_MSG_ID, menedżer kolejek generuje unikalny identyfikator komunikatu.[3]po umieszczeniu komunikatu i umieszczeniu go w deskryptorze komunikatu wysłanym razem z komunikatem. Menedżer kolejek zwraca również ten identyfikator komunikatu w deskryptorze komunikatu należącym do aplikacji wysyłającej. Aplikacja może użyć tej wartości do rejestrowania informacji o konkretnych komunikatach i do odpowiadania na zapytania z innych części aplikacji.

Jeśli komunikat jest umieszczany w temacie, menedżer kolejek generuje unikalne identyfikatory komunikatów zgodnie z potrzebami dla każdego publikowanego komunikatu. Jeśli parametr MQPMO_NEW_MSG_ID jest określony przez aplikację, menedżer kolejek generuje unikalny identyfikator komunikatu, który ma być zwracany na wyjściu. Jeśli parametr MQMI_NONE jest określony przez aplikację, wartość pola *MsgId* w deskryptorze MQMD jest niezmieniona po powrocie z wywołania.

Więcej informacji na temat zachowanych publikacji zawiera opis komendy MQPMO_RETAIN w sekcji “Opcje (MQLONG) dla MQPMO” na stronie 520 .

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, menedżer kolejek w razie potrzeby generuje unikalne identyfikatory komunikatów, ale wartość pola *MsgId* w strukturze MQMD pozostaje niezmieniona po powrocie z wywołania, nawet jeśli określono opcję MQMI_NONE lub MQPMO_NEW_MSG_ID. Jeśli aplikacja musi znać identyfikatory komunikatów wygenerowane przez menedżer kolejek, musi udostępnić rekordy MQPMR zawierające pole *MsgId* .

Aplikacja wysyłająca może również określić wartość identyfikatora komunikatu inną niż MQMI_NONE; spowoduje to zatrzymanie menedżera kolejek generującego unikalny identyfikator komunikatu. Aplikacja, która przekazuje komunikat, może użyć tej opcji do propagowania identyfikatora oryginalnego komunikatu.

Menedżer kolejek nie używa tego pola z wyjątkiem następujących:

- Generuj unikalną wartość na żądanie, zgodnie z powyższym opisem
- Dostarcz wartość do aplikacji, która wysyła żądanie pobrania komunikatu

[3] Kod *MsgId* wygenerowany przez menedżera kolejek składa się z 4-bajtowego identyfikatora produktu (AMQ¬ lub CSQ¬ w kodzie ASCII lub EBCDIC, gdzie ¬ reprezentuje znak odstępu), po którym następuje specyficzna dla produktu implementacja unikalnego łańcucha. W pliku IBM MQ zawiera on pierwsze 12 znaków nazwy menedżera kolejek i wartość pochodzącą z zegara systemowego. Wszystkie menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się pierwszymi 12 znakami, aby identyfikatory komunikatów były unikalne. Możliwość generowania unikalnego łańcucha zależy również od tego, czy zegar systemowy nie jest zmieniany wstecz. Aby wyeliminować możliwość, że identyfikator komunikatu wygenerowany przez menedżer kolejek duplikuje identyfikator wygenerowany przez aplikację, aplikacja musi unikać generowania identyfikatorów ze znakami początkowymi z zakresu od A do I w kodzie ASCII lub EBCDIC (X'41 'do X'49' i X'C1'do X'C9'). Jednak aplikacja nie może generować identyfikatorów z początkowymi znakami w tych zakresach.

- Skopiuj wartość do pola *CorrelId* dowolnego komunikatu raportu, który jest generowany dla tego komunikatu (w zależności od opcji *Report*).

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole *MsgId* w sposób określony w polu *Report* oryginalnego komunikatu (MQRO_NEW_MSG_ID lub MQRO_PASS_MSG_ID). W tym celu muszą być również używane aplikacje generujące komunikaty raportów.

W przypadku wywołania MQGET *MsgId* jest jednym z pięciu pól, którego można użyć do pobrania konkretnego komunikatu z kolejki. Zwykle wywołanie MQGET zwraca następny komunikat w kolejce, ale konkretny komunikat można uzyskać, określając co najmniej jedno z pięciu kryteriów wyboru w dowolnej kombinacji. Są to następujące pola:

- *MsgId*
- *CorrelId*
- *GroupId*
- *MsgSeqNumber*
- *Offset*

Aplikacja ustawia co najmniej jedno z tych pól na wymagane wartości, a następnie ustawia odpowiednie opcje zgodności MQMO_ * w polu *MatchOptions* w MQGMO w celu użycia tych pól jako kryteriów wyboru. Tylko komunikaty, które mają określone wartości w tych polach, są kandydatami do pobrania. Wartość domyślna dla pola *MatchOptions* (jeśli nie została zmieniona przez aplikację) jest zgodna zarówno z identyfikatorem komunikatu, jak i identyfikatorem korelacji.

W systemie z/OSkryteria wyboru, których można użyć, są ograniczone przez typ indeksu używanego dla kolejki. Więcej informacji na ten temat zawiera opis atrybutu kolejki **IndexType** .

Zwykle zwracany jest *pierwszy* komunikat w kolejce, który spełnia kryteria wyboru. Jeśli jednak określono opcję MQGMO_BROWSE_NEXT, zwracany jest komunikat *następny* , który spełnia kryteria wyboru. Skanowanie tego komunikatu rozpoczyna się od komunikatu *następującego po* bieżącej pozycji kursora.

Uwaga: Kolejka jest skanowana sekwencyjnie w poszukiwaniu komunikatu spełniającego kryteria wyboru, dlatego czasy pobierania są dłuższe niż w przypadku braku określonych kryteriów wyboru, zwłaszcza jeśli przed znalezniem odpowiedniego komunikatu należy przeskanować wiele komunikatów. Wyjątki od tej zasady są następujące:

- Multi Wywołanie MQGET przez program *CorrelId* w 64-bitowym środowisku wieloplatformowym, w którym indeks *CorrelId* eliminuje potrzebę wykonywania prawdziwego sekwencyjnego skanowania.
- z/OS Wywołanie MQGET przez komendę *IndexType* w systemie z/OS.

W obu tych przypadkach wydajność pobierania jest większa.

Więcej informacji na temat używania kryteriów wyboru w różnych sytuacjach zawiera sekcja Tabela 495 na stronie 403 .

Podanie wartości MQMI_NONE jako identyfikatora komunikatu ma taki sam skutek, jak niepodanie wartości MQMO_MATCH_MSG_ID, czyli *dowolnego* zgodnego identyfikatora komunikatu.

To pole jest ignorowane, jeśli w parametrze **GetMsgOpts** wywołania MQGET określono opcję MQGMO_MSG_UNDER_CURSOR.

W przypadku powrotu z wywołania MQGET pole *MsgId* jest ustawiane na identyfikator zwróconego komunikatu (jeśli istnieje).

Można użyć następującej wartości specjalnej:

MQMI_BRAK
Nie określono identyfikatora komunikatu.

Wartością długości pola jest zero binarne.

W języku programowania C zdefiniowana jest również stała MQMI_NONE_ARRAY. Ma ona taką samą wartość jak MQMI_NONE, ale jest tablicą znaków, a nie łańcuchem.

Jest to pole wejściowe/wyjściowe dla wywołań MQGET, MQPUT i MQPUT1 . Długość tego pola jest określona przez wartość MQ_MSG_ID_LENGTH. Wartością początkową tego pola jest MQMI_NONE.

CorrelId (MQBYTE24) dla MQMD

Pole CorrelId jest właściwością w nagłówku komunikatu, która może być używana do identyfikowania konkretnego komunikatu lub grupy komunikatów.

Jest to łańcuch bajtów, który może zostać użyty przez aplikację do powiązania jednego komunikatu z innym lub do powiązania komunikatu z inną pracą wykonywaną przez aplikację. Identyfikator korelacji jest trwałą właściwością komunikatu i jest zachowywany po restarcie menedżera kolejek. Ponieważ identyfikator korelacji jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator korelacji nie jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do innego.

W przypadku wywołań MQPUT i MQPUT1 aplikacja może określić dowolną wartość. Menedżer kolejek przesyła tę wartość wraz z komunikatem i dostarcza ją do aplikacji, która wysyła żądanie pobrania komunikatu.

Jeśli aplikacja określa identyfikator MQPMO_NEW_CORREL_ID, menedżer kolejek generuje unikalny identyfikator korelacji, który jest wysyłany z komunikatem, a także zwracany do aplikacji wysyłającej w wyniku wywołania MQPUT lub MQPUT1 .

Identyfikator korelacji wygenerowany przez menedżer kolejek składa się z 3-bajtowego identyfikatora produktu (AMQ lub CSQ w kodzie ASCII lub EBCDIC), po którym następuje jeden zarezerwowany bajt i specyficzna dla produktu implementacja unikalnego łańcucha. W produkcie IBM MQ ten specyficzny dla produktu łańcuch implementacji zawiera pierwsze 12 znaków nazwy menedżera kolejek i wartość pochodzącą z zegara systemowego. Wszystkie menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się pierwszymi 12 znakami, aby identyfikatory komunikatów były unikalne. Możliwość generowania unikalnego łańcucha zależy również od tego, czy zegar systemowy nie jest zmieniany wstecz. Aby wyeliminować możliwość, że identyfikator komunikatu wygenerowany przez menedżer kolejek duplikuje identyfikator wygenerowany przez aplikację, aplikacja musi unikać generowania identyfikatorów ze znakami początkowymi z zakresu od A do I w kodzie ASCII lub EBCDIC (X'41 'do X'49' i X'C1'do X'C9'). Jednak aplikacja nie może generować identyfikatorów z początkowymi znakami w tych zakresach.

Ten wygenerowany identyfikator korelacji jest zachowywany wraz z komunikatem (jeśli został zachowany) i jest używany jako identyfikator korelacji, gdy komunikat jest wysyłany jako publikacja do subskrybentów, którzy w polu SubCorrelId w MQSD przekazanym w wywołaniu MQSUB podali wartość MQCI_NONE. Więcej informacji na temat zachowanych publikacji zawiera sekcja Opcje MQPMO .

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole *CorrelId* w sposób określony w polu *Report* oryginalnego komunikatu: MQRO_COPY_MSG_ID_TO_CORREL_ID lub MQRO_PASS_CORREL_ID. W tym celu muszą być również używane aplikacje generujące komunikaty raportów.

W przypadku wywołania MQGET *CorrelId* jest jednym z pięciu pól, za pomocą których można wybrać konkretny komunikat do pobrania z kolejki. Szczegółowe informacje na temat określania wartości dla tego pola zawiera opis pola *MsgId* .

Podanie wartości MQCI_NONE jako identyfikatora korelacji ma taki sam efekt, jak niepodanie wartości MQMO_MATCH_CORREL_ID, czyli *dowolny* identyfikator korelacji będzie zgodny.

Jeśli opcja MQGMO_MSG_UNDER_CURSOR jest określona w parametrze **GetMsgOpts** wywołania MQGET, to pole jest ignorowane.

Po powrocie z wywołania MQGET pole *CorrelId* jest ustawiane na identyfikator korelacji zwróconego komunikatu (jeśli istnieje).

Można użyć następujących wartości specjalnych:

MQCI_BRAK
Nie określono identyfikatora korelacji.

Wartością długości pola jest zero binarne.

Dla języka programowania C zdefiniowana jest również stała MQCI_NONE_ARRAY. Ma ona taką samą wartość jak parametr MQCI_NONE, ale jest tablicą znaków zamiast łańcucha.

MQCI_NEW_SESSION
Komunikat jest początkiem nowej sesji.

Ta wartość jest rozpoznawana przez CICS bridge jako początek nowej sesji, czyli początek nowej sekwencji komunikatów.

Dla języka programowania C zdefiniowana jest również stała MQCI_NEW_SESSION_ARRAY. Ma ona taką samą wartość jak parametr MQCI_NEW_SESSION, ale jest tablicą znaków zamiast łańcucha.

W przypadku wywołania MQGET jest to pole wejściowe/wyjściowe. W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe, jeśli nie określono MQPMO_NEW_CORREL_ID, lub pole wyjściowe, jeśli określono MQPMO_NEW_CORREL_ID. Długość tego pola jest określona przez wartość MQ_CORREL_ID_LENGTH. Wartością początkową tego pola jest MQCI_NONE.

Uwaga:

Nie można przekazać identyfikatora korelacji publikacji w hierarchii. Pole jest używane przez menedżer kolejek.

BackoutCount (MQLONG) dla MQMD

Jest to liczba wskazująca, ile razy komunikat został poprzednio zwrócony przez wywołanie MQGET w ramach jednostki pracy, a następnie wycofany. Pomaga on aplikacji w wykrywaniu błędów przetwarzania, które są oparte na treści komunikatu. Ta liczba wyklucza wywołania MQGET, które określają dowolną z opcji MQGMO_BROWSE_ *.

Na dokładność tej liczby ma wpływ atrybut kolejki **HardenGetBackout** ; patrz sekcja "Atrybuty kolejek" na stronie 863.

W systemie z/OSwartość 255 oznacza, że komunikat został wycofany 255 lub więcej razy; zwracana wartość nigdy nie jest większa niż 255.

Jest to pole wyjściowe wywołania MQGET. Jest on ignorowany w przypadku wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest 0.

ReplyToQ (MQCHAR48) dla MQMD

Jest to nazwa kolejki komunikatów, do której aplikacja, która wysłała żądanie pobrania komunikatu, wysyła komunikaty MQMT_REPLY i MQMT_REPORT. Nazwa jest nazwą lokalną kolejki, która jest zdefiniowana w menedżerze kolejek identyfikowanym przez *ReplyToQMgr*. Ta kolejka nie może być kolejką modelową, chociaż nadawczy menedżer kolejek nie sprawdza tego podczas umieszczania komunikatu.

W przypadku wywołań MQPUT i MQPUT1 to pole nie może być puste, jeśli pole *MsgType* ma wartość MQMT_REQUEST lub jeśli jakieś komunikaty raportu są żądane przez pole *Report* . Jednak określona (lub podstawiona) wartość jest przekazywana do aplikacji, która wysyła żądanie pobrania dla komunikatu, niezależnie od typu komunikatu.

Jeśli pole *ReplyToQMgr* jest puste, menedżer kolejek lokalnych wyszukuje nazwę *ReplyToQ* we własnych definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość *ReplyToQ* w przesłanym komunikacie jest zastępowana wartością atrybutu **RemoteQName** z definicji kolejki zdalnej i ta wartość jest zwracana w deskryptorze komunikatu, gdy aplikacja odbierająca wysyła wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, wartość *ReplyToQ* pozostaje niezmieniona.

Jeśli nazwa jest określona, może zawierać końcowe odstępy; pierwszy znak o kodzie zero i następujące po nim znaki są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane sprawdzanie, czy

nazwa jest zgodna z regułami nazewnictwa dla kolejek. Jest to również prawdą dla przesyłanej nazwy, jeśli nazwa *ReplyToQ* jest zastępowana w przesłanym komunikacie. Jedynym sprawdzonym sposobem jest określenie nazwy, jeśli okoliczności tego wymagają.

Jeśli kolejka odpowiedzi nie jest wymagana, należy ustawić pole *ReplyToQ* na wartość pustą lub (w języku programowania C) na łańcuch pusty albo na co najmniej jedną wartość pustą, po której następuje znak pusty. Nie należy pozostawiać pola niezainicjowanego.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnianą spacjami do długości pola.

Jeśli nie można dostarczyć komunikatu, który wymaga komunikatu raportu, a komunikat raportu również nie może zostać dostarczony do określonej kolejki, zarówno oryginalny komunikat, jak i komunikat raportu są umieszczane w kolejce niedostarczonych komunikatów (patrz atrybut **DeadLetterQName** opisany w sekcji "Atrybuty menedżera kolejek" na stronie 824).

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

Menedżer kolejek ReplyTo(MQCHAR48) dla MQMD

Jest to nazwa menedżera kolejek, do którego ma zostać wysłany komunikat odpowiedzi lub komunikat raportu. *ReplyToQ* to nazwa lokalna kolejki, która jest zdefiniowana w tym menedżerze kolejek.

Jeśli pole *ReplyToQMgr* jest puste, menedżer kolejek lokalnych wyszukuje nazwę *ReplyToQ* w swoich definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość *ReplyToQMgr* w przesłanym komunikacie jest zastępowana wartością atrybutu **RemoteQMgrName** z definicji kolejki zdalnej i ta wartość jest zwracana w deskryptorze komunikatu, gdy aplikacja odbierająca wysyła wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, *ReplyToQMgr* przesyłany z komunikatem jest nazwą lokalnego menedżera kolejek.

Jeśli nazwa jest określona, może zawierać końcowe odstępy; pierwszy znak o kodzie zero i następujące po nim znaki są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane sprawdzanie, czy nazwa jest zgodna z regułami nazewnictwa menedżerów kolejek lub czy ta nazwa jest rozpoznawana przez nadawczy menedżer kolejek. Ta sytuacja ma również zastosowanie w przypadku przesłanej nazwy, jeśli nazwa *ReplyToQMgr* jest zastępowana w przesłanym komunikacie.

Jeśli kolejka odpowiedzi nie jest wymagana, należy ustawić pole *ReplyToQMgr* na wartość pustą lub (w języku programowania C) na łańcuch pusty albo na co najmniej jedną wartość pustą, po której następuje znak pusty. Nie należy pozostawiać pola niezainicjowanego.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnianą spacjami do długości pola.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

UserIdentifier (MQCHAR12) dla deskryptora MQMD

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja "MQMD-deskryptor komunikatu" na stronie 432 i sekcja Kontekst komunikatu.

UserIdentifier określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu.

Po odebraniu komunikatu należy użyć pola *UserIdentifier* w polu *AlternateUserId* parametru **ObjDesc** kolejnego wywołania MQOPEN lub MQPUT1 , aby wykonać sprawdzenie autoryzacji dla użytkownika *UserIdentifier* zamiast aplikacji wykonującej operację otwierania.

Gdy menedżer kolejek generuje te informacje dla wywołania MQPUT lub MQPUT1 :

- W systemie z/OSmenedżer kolejek używa parametru *AlternateUserId* from **ObjDesc** wywołania MQOPEN lub MQPUT1 , jeśli określono opcję MQOO_ALTERNATE_USER_AUTHORITY lub

MQPMO_ALTERNATE_USER_AUTHORITY. Jeśli odpowiednia opcja nie została określona, menedżer kolejek używa identyfikatora użytkownika określonego na podstawie środowiska.

- W innych środowiskach menedżer kolejek zawsze używa identyfikatora użytkownika określonego na podstawie środowiska.

Jeśli identyfikator użytkownika jest określany na podstawie środowiska:

- W systemie z/OSmenedżer kolejek używa:
  - W przypadku systemu MVS (zadanie wsadowe): identyfikator użytkownika z karty JES JOB lub uruchomionego zadania
  - W przypadku TSO identyfikator użytkownika propagowany do zadania podczas wprowadzania zadania
  - W przypadku systemu CICSjest to identyfikator użytkownika powiązany z zadaniem.
  - W przypadku systemu IMSidentyfikator użytkownika zależy od typu aplikacji:
    - Przez:
      - Regiony BMP bez komunikatów
      - Regiony IFP niebędące wiadomością
      - Regiony BMP komunikatu i IFP komunikatu, które nie wywołały pomyślnego wywołania GU

      Menedżer kolejek używa identyfikatora użytkownika z karty JES JOB regionu lub identyfikatora użytkownika TSO. Jeśli są one puste lub mają wartość null, używana jest nazwa bloku specyfikacji programu (PSB).
    - Przez:
      - Regiony BMP komunikatów i IFP komunikatów, które *mają* pomyślne wywołanie GU
      - Regiony MPP

      menedżer kolejek używa jednej z następujących wartości:
      - Identyfikator zalogowanego użytkownika powiązany z komunikatem
      - Nazwa terminalu logicznego (LTERM)
      - Identyfikator użytkownika z karty JES JOB regionu
      - Identyfikator użytkownika TSO
      - Nazwa PSB
- W systemie IBM imenedżer kolejek używa nazwy profilu użytkownika powiązanego z zadaniem aplikacji.
- W systemie AIX and Linuxmenedżer kolejek używa:
  - Nazwa logowania aplikacji
  - Efektywny identyfikator użytkownika procesu, jeśli nie jest dostępne logowanie
  - Identyfikator użytkownika powiązany z transakcją, jeśli aplikacja jest transakcją CICS
- W systemach Windows menedżer kolejek używa pierwszych 12 znaków nazwy zalogowanego użytkownika.

To pole jest zwykle polem wyjściowym wygenerowanym przez menedżer kolejek, ale w przypadku wywołania MQPUT lub MQPUT1 można ustawić to pole jako pole wejściowe/wyjściowe i określić pole UserIdentification zamiast pozwalać menedżerowi kolejek na generowanie tych informacji. Określ wartość MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT w parametrze PutMsgOpts i podaj identyfikator użytkownika w polu UserIdentifier , jeśli menedżer kolejek nie ma generować pola UserIdentifier dla wywołania MQPUT lub MQPUT1 .

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT. Wszelkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące po nim znaki w odstępy. Jeśli nie określono opcji MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość *UserIdentifier* , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru *UserIdentifier* , która jest zachowywana razem z komunikatem, jeśli zostanie zachowany (więcej informacji na temat zachowanych publikacji zawiera opis parametru MQPMO_RETAIN), ale nie jest używana jako parametr *UserIdentifier* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępnia on wartość przesłaniającą parametr *UserIdentifier* we wszystkich wysłanych do nich publikacjach. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 12 znaków odstępu w innych językach programowania.

AccountingToken (MQBYTE32) dla MQMD

Jest to token rozliczania, część *kontekstu tożsamości* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432 i sekcja Kontekst komunikatu.

Produkt `AccountingToken` umożliwia aplikacji odpowiednie naliczanie opłat za pracę wykonaną w wyniku komunikatu. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza ich treści.

Menedżer kolejek generuje te informacje w następujący sposób:

- Pierwszy bajt pola jest ustawiony na długość informacji rozliczeniowych obecnych w kolejnych bajtach. Długość ta jest z zakresu od 0 do 30 i jest przechowywana w pierwszym bajcie jako binarna liczba całkowita.
- Drugi i kolejne bajty (określone w polu długości) są ustawione na informacje rozliczeniowe odpowiednie dla środowiska.
  - z/OS W systemie z/OS informacje rozliczeniowe są ustawione na:
    - W przypadku zadania wsadowego z/OS informacje rozliczeniowe z karty JES JOB lub z instrukcji JES ACCT na karcie EXEC (separatory przecinków są zmieniane na X'FF '). W razie potrzeby informacje te są obcinane do 31 bajtów.
    - W przypadku TSO jest to numer konta użytkownika.
    - W przypadku systemu CICSjest to identyfikator jednostki pracy jednostki logicznej 6.2 (UEPUOWDS) (26 bajtów).
    - W systemie IMS: 8-znakowa nazwa PSB połączona z 16-znakowym tokenem odtwarzania IMS .
  - IBM i W systemie IBM iinformacje rozliczeniowe są ustawiane na kod rozliczeniowy zadania.
  - Linux AIX W systemie AIX and Linuxinformacje rozliczeniowe są ustawiane na liczbowy identyfikator użytkownika w postaci znaków ASCII.
  - Windows W systemie Windowsinformacje rozliczeniowe są ustawiane na identyfikator bezpieczeństwa systemu Windows (SID) w formacie skompresowanym. Identyfikator SID jednoznacznie identyfikuje identyfikator użytkownika zapisany w polu *UserIdentifier* . Jeśli identyfikator SID jest przechowywany w polu *AccountingToken* , pomijane jest 6-bajtowe uprawnienie identyfikatora (znajdujące się w trzecim i kolejnych bajtach identyfikatora SID). Na przykład, jeśli identyfikator SID Windows ma długość 28 bajtów, w polu *AccountingToken* zostaną zapisane 22 bajty informacji o identyfikatorze SID.
- Ostatni bajt (bajt 32) pola rozliczania jest ustawiony na typ tokenu rozliczania (w tym przypadku MQACTT_NT_SECURITY_ID, x '0b'):

  MQACTT_CICS_LUOW_ID
  CICS Identyfikator LUOW.

  Windows **MQACTT_NT_SECURITY_ID**
  Identyfikator zabezpieczeń Windows .

IBM i **MQACTT_OS400_ACCOUNT_TOKEN**
IBM i token rozliczania.

UNIX **MQACTT_UNIX_NUMERIC_ID**
UNIX identyfikator liczbowy.

Użytkownik MQACTT_USER
Token rozliczania zdefiniowany przez użytkownika.

MQACTT_UNKNOWN
Nieznany typ tokenu rozliczania.

Typ tokenu rozliczania jest ustawiany na wartość jawną tylko w następujących środowiskach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows

i dla IBM MQ MQI clients połączonych z tymi systemami. W innych środowiskach typ znacznika rozliczania jest ustawiany na wartość MQACTT_UNKNOWN. W tych środowiskach należy użyć pola *PutApplType* , aby określić typ odebranego tokenu rozliczeniowego.

- Wszystkie pozostałe bajty są ustawione na zero binarne.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT. Jeśli nie określono ani parametru MQPMO_SET_IDENTITY_CONTEXT, ani parametru MQPMO_SET_ALL_CONTEXT, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość `AccountingToken` , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru `AccountingToken` , która jest przechowywana razem z komunikatem, jeśli został zachowany (więcej informacji na temat zachowanych publikacji zawiera opis parametru MQPMO_RETAIN w sekcji “Opcje (MQLONG) dla MQPMO” na stronie 520), ale nie jest używana jako parametr `AccountingToken` , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępnia on wartość przesłaniającą parametr `AccountingToken` we wszystkich wysłanych do nich publikacjach. Jeśli komunikat nie ma kontekstu, pole jest całkowicie binarne zero.

Jest to pole wyjściowe wywołania MQGET.

To pole nie podlega żadnej translacji na podstawie zestawu znaków menedżera kolejek. Pole jest traktowane jako łańcuch bitów, a nie jako łańcuch znaków.

Menedżer kolejek nie wykonuje żadnych działań z informacjami w tym polu. Aplikacja musi interpretować informacje, jeśli chce je wykorzystać do celów księgowych.

W polu `AccountingToken` można użyć następującej wartości specjalnej:

MQACT_NONE
Nie określono tokenu rozliczania.

Wartością długości pola jest zero binarne.

Dla języka programowania C zdefiniowana jest również stała MQACT_NONE_ARRAY, która ma taką samą wartość jak MQACT_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest określona przez parametr MQ_ACCOUNTING_TOKEN_LENGTH. Wartością początkową tego pola jest MQACT_NONE.

Dane ApplIdentity(MQCHAR32) dla MQMD

Jest to część **kontekstu tożsamości** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja "MQMD-deskryptor komunikatu" na stronie 432 i sekcja Kontekst komunikatu.

ApplIdentityData to informacje, które są definiowane przez pakiet aplikacji i mogą być używane do udostępniania dodatkowych informacji o komunikacie lub jego twórcy. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu. Gdy menedżer kolejek generuje te informacje, jest całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT. Jeśli występuje znak o kodzie zero, menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące po nim znaki w znaki puste. Jeśli nie określono ani parametru MQPMO_SET_IDENTITY_CONTEXT, ani parametru MQPMO_SET_ALL_CONTEXT, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość *ApplIdentityData* , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru *ApplIdentityData* , która jest zachowywana razem z komunikatem, jeśli zostanie zachowany (więcej informacji na temat zachowanych publikacji zawiera opis parametru MQPMO_RETAIN), ale nie jest używana jako parametr *ApplIdentityData* , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępnia on wartość przesłaniającą parametr *ApplIdentityData* we wszystkich wysłanych do nich publikacjach. Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez wartość MQ_APPL_IDENTITY_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 32 znaki odstępu w innych językach programowania.

PutApplTyp (MQLONG) dla MQMD

Jest to typ aplikacji, która umieściła komunikat i jest częścią **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja "MQMD-deskryptor komunikatu" na stronie 432 i sekcja Kontekst komunikatu.

PutApplType może mieć jeden z następujących typów standardowych. Można również zdefiniować własne typy, ale tylko z wartościami z zakresu od MQAT_USER_FIRST do MQAT_USER_LAST.

MQAT_AIX
Aplikacja AIX (taka sama jak MQAT_UNIX).

MQAT_AMQP
Aplikacja protokołu AMQP

MQAT_BROKER
Broker.

MQAT_CICS
CICS .

MQAT_CICS_BRIDGE
CICS bridge.

MQAT_CICS_VSE
CICS/VSE .

MQAT_DOS
Aplikacja IBM MQ MQI client w systemie DOS na komputerze PC.

MQAT_DQM
Agent rozproszonego menedżera kolejek.

MQAT_GUARDIAN (strażnik MQ)
Aplikacja Tandem Guardian (taka sama wartość jak MQAT_NSK).

MQAT_IMS
Aplikacja IMS .

MQAT_IMS_BRIDGE,
Most IMS .

MQAT_JAVA
Java.

MQAT_MVS
Aplikacja MVS lub TSO (taka sama wartość jak MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Aplikacja agenta.

MQAT_OS390
Aplikacja OS/390 (taka sama jak MQAT_ZOS).

MQAT_OS400
Aplikacja IBM i .

MQAT_QMGR
menedżerze kolejek.

MQAT_UNIX
Aplikacja UNIX .

MQAT_VOS
Aplikacja Stratus VOS.

MQAT_WINDOWS
16-bitowa aplikacja Windows .

MQAT_WINDOWS_NT
32-bitowa aplikacja Windows .

MQAT_WLM
Aplikacja menedżera obciążenia z/OS .

MQAT_XCF,
XCF.

MQAT_ZOS
Aplikacja z/OS .

MQAT_DEFAULT
Domyślny typ aplikacji.

Jest to domyślny typ aplikacji dla platformy, na której działa aplikacja.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska. Z tego powodu należy zawsze kompilować aplikację przy użyciu plików nagłówkowych, dołączanych lub COPY, które są odpowiednie dla platformy, na której aplikacja będzie uruchamiana.

MQAT_UNKNOWN
Ta wartość wskazuje, że typ aplikacji jest nieznany, nawet jeśli istnieją inne informacje o kontekście.

MQAT_USER_FIRST
Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Tabela MQAT_USER_LAST
Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Może również wystąpić następująca wartość specjalna:

MQAT_NO_CONTEXT
Ta wartość jest ustawiana przez menedżer kolejek, gdy komunikat jest umieszczany bez kontekstu (oznacza to, że określono opcję kontekstu MQPMO_NO_CONTEXT).

Po pobraniu komunikatu produkt *PutApplType* może zostać przetestowany pod kątem tej wartości w celu określenia, czy komunikat ma kontekst (zaleca się, aby parametr *PutApplType* nie był nigdy ustawiany na wartość MQAT_NO_CONTEXT przez aplikację używającą parametru MQPMO_SET_ALL_CONTEXT, jeśli jakiekolwiek inne pola kontekstu nie są puste).

Gdy menedżer kolejek generuje te informacje w wyniku umieszczenia w aplikacji, pole jest ustawiane na wartość, która jest określana przez środowisko. W systemie IBM ijest on ustawiony na wartość MQAT_OS400; menedżer kolejek nigdy nie używa programu MQAT_CICS w systemie IBM i.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_ALL_CONTEXT. Jeśli opcja MQPMO_SET_ALL_CONTEXT nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Jest to pole wyjściowe wywołania MQGET. Wartością początkową tego pola jest MQAT_NO_CONTEXT.

PutApplNazwa (MQCHAR28) dla MQMD

Jest to nazwa aplikacji, która umieściła komunikat i jest częścią *kontekstu źródłowego* komunikatu. Zawartość jest różna dla różnych platform i może być różna dla różnych wersji.

Więcej informacji na temat kontekstu komunikatu zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432 i sekcja Kontekst komunikatu.

W programie IBM MQ 9.1.2 można określić nazwę aplikacji w dodatkowych językach programowania. Więcej informacji na ten temat zawiera sekcja określanie nazwy aplikacji w obsługiwanych językach programowania .

Format zmiennej *PutApplName* zależy od wartości zmiennej *PutApplType* i może zmieniać się w zależności od wersji. Zmiany są rzadkie, ale należy je wprowadzić w przypadku zmiany środowiska.

Gdy menedżer kolejek ustawia to pole (czyli dla wszystkich opcji z wyjątkiem opcji MQPMO_SET_ALL_CONTEXT), ustawia to pole na wartość określoną przez środowisko:

- z/OS W systemie z/OSmenedżer kolejek używa:
  - W przypadku zadania wsadowego z/OS jest to 8-znakowa nazwa zadania z karty JES JOB
  - W przypadku TSO-7-znakowy identyfikator użytkownika TSO
  - W systemie CICS: 8-znakowa zmienna applid, po której następuje 4-znakowa zmienna tranid
  - W systemie IMS: 8-znakowy identyfikator systemu IMS , po którym następuje 8-znakowa nazwa PSB
  - Dla XCF: 8-znakowa nazwa grupy XCF, po której następuje 16-znakowa nazwa podzbioru XCF
  - W przypadku komunikatu wygenerowanego przez menedżer kolejek pierwsze 28 znaków nazwy menedżera kolejek
  - W przypadku kolejkowania rozproszonego bez systemu CICS: 8-znakowa nazwa zadania inicjatora kanału, po której następuje 8-znakowa nazwa modułu umieszczającego w kolejce niedostarczonych komunikatów, po której następuje 8-znakowy identyfikator zadania.

  Nazwa lub nazwy są dopełniane po prawej stronie odstępami, tak jak każda spacja w pozostałej części pola. Jeśli istnieje więcej niż jedna nazwa, nie ma między nimi separatora.
- Windows W systemach Windows menedżer kolejek używa następujących nazw:
  - W przypadku aplikacji CICS jest to nazwa transakcji CICS
  - W przypadku aplikacji innej niż aplikacjaCICS , 28 znaków po prawej stronie pełnej nazwy pliku wykonywalnego
- IBM i W systemie IBM imenedżer kolejek używa pełnej nazwy zadania.
- Linux AIX W systemie AIX and Linuxmenedżer kolejek używa następujących nazw:
  - W przypadku aplikacji CICS jest to nazwa transakcji CICS
  - W przypadku aplikacji innej niżCICS produkt MQ pyta system operacyjny o nazwę procesu. Jest ona zwracana jako nazwa pliku programu bez pełnej ścieżki. Następnie produkt MQ umieszcza tę nazwę procesu w strukturze MQMD.PutApplName :

    AIX **AIX**

    Jeśli nazwa jest mniejsza lub równa 28 bajtom, nazwa jest wstawiana i uzupełniana po prawej stronie spacjami.

Jeśli nazwa jest dłuższa niż 28 bajtów, wstawianych jest 28 bajtów z lewej strony nazwy.

Linux

Jeśli nazwa jest mniejsza lub równa 15 bajtom, nazwa jest wstawiana, dopełniona spacjami z prawej strony.

Jeśli nazwa jest dłuższa niż 15 bajtów, to po lewej stronie wstawiane jest 15 bajtów nazwy, dopełniane spacjami z prawej strony.

Na przykład w przypadku uruchomienia komendy `/opt/mqm/samp/bin/amqsput QNAME` QMNAMEnazwą PutAppljest `'amqsput'`. W tym polu MQCHAR28 znajduje się 21 spacji. Należy zauważyć, że pełna ścieżka zawierająca `/opt/mqm/samp/bin` nie jest uwzględniana w nazwie PutAppl.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_ALL_CONTEXT. Wszelkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Znak o kodzie zero i wszystkie następujące po nim znaki są przekształcane przez menedżera kolejek w odstępy. Jeśli opcja MQPMO_SET_ALL_CONTEXT nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

PutDate (MQCHAR8) dla MQMD

Jest to data umieszczenia komunikatu i jest ona częścią **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432 i sekcja Kontekst komunikatu.

Format używany dla daty wygenerowania tego pola przez menedżer kolejek to:

- RRRRMMDD

gdzie znaki reprezentują:

rrrr
rok (cztery cyfry)

MM
miesiąc roku (od 01 do 12)

DD
dzień miesiąca (od 01 do 31)

Czas Greenwich (Greenwich Mean Time-GMT) jest używany w polach *PutDate* i *PutTime* , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, datą jest data umieszczenia komunikatu, a nie data zatwierdzenia jednostki pracy.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_ALL_CONTEXT. Zawartość pola nie jest sprawdzana przez menedżer kolejek, z wyjątkiem tego, że wszystkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące po nim znaki w odstępy. Jeśli opcja MQPMO_SET_ALL_CONTEXT nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez wartość MQ_PUT_DATE_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 8 znaków odstępu w innych językach programowania.

PutTime (MQCHAR8) dla MQMD

Jest to czas umieszczenia komunikatu i jest on częścią **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432 i sekcja Kontekst komunikatu.

Format używany dla czasu wygenerowania tego pola przez menedżer kolejek to:

- GGMMSSTH

gdzie znaki reprezentują (w kolejności):

GG
godzin (od 00 do 23)

MM
minuty (od 00 do 59)

SS
sekundy (od 00 do 59; patrz uwaga)

T
dziesiąte części sekundy (od 0 do 9)

H
setne sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z bardzo dokładnym standardem czasu, w rzadkich przypadkach możliwe jest zwrócenie wartości 60 lub 61 dla sekund w produkcie *PutTime*. Dzieje się tak, gdy sekundy przestępne są wstawiane do globalnego standardu czasu.

Czas Greenwich (Greenwich Mean Time-GMT) jest używany w polach *PutDate* i *PutTime* , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, jest to czas umieszczenia komunikatu, a nie czas zatwierdzenia jednostki pracy.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_ALL_CONTEXT. Menedżer kolejek nie sprawdza zawartości pola, z wyjątkiem tego, że wszystkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące po nim znaki w odstępy. Jeśli opcja MQPMO_SET_ALL_CONTEXT nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez wartość MQ_PUT_TIME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 8 znaków odstępu w innych językach programowania.

ApplOriginData (MQCHAR4) dla MQMD

Jest to część *kontekstu źródłowego* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432 i sekcja Kontekst komunikatu.

`ApplOriginData` to informacje zdefiniowane przez pakiet aplikacji, których można użyć do udostępnienia dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiana przez aplikacje działające z odpowiednimi uprawnieniami użytkownika w celu wskazania, czy dane tożsamości są zaufane.

Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu. Gdy menedżer kolejek generuje te informacje, jest całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PutMsgOpts** określono wartość MQPMO_SET_ALL_CONTEXT. Wszelkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące po nim znaki w odstępy. Jeśli opcja MQPMO_SET_ALL_CONTEXT nie jest określona, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez wartość MQ_APPL_ORIGIN_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 4 znaki odstępu w innych językach programowania.

Po opublikowaniu komunikatu, mimo że parametr `ApplOriginData` jest ustawiony, jest on pusty w subskrypcji, która jest przez niego odbierana.

GroupId (MQBYTE24) dla MQMD

Jest to łańcuch bajtów używany do identyfikowania konkretnej grupy komunikatów lub komunikatu logicznego, do którego należy komunikat fizyczny. Parametr *GroupId* jest również używany, jeśli dla komunikatu dozwolona jest segmentacja. We wszystkich tych przypadkach *GroupId* ma wartość inną niż NULL i w polu *MsgFlags* jest ustawiona co najmniej jedna z następujących opcji:

- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_SEGMENTATION_ALLOWED

Jeśli żadna z tych opcji nie jest ustawiona, *GroupId* ma specjalną wartość NULL MQGI_NONE.

Aplikacja nie musi ustawiać tego pola w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono parametr MQPMO_LOGICAL_ORDER.
- W wywołaniu MQGET nie określono wartości MQMO_MATCH_GROUP_ID.

Są to zalecane sposoby używania tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołanie ma wartość MQPUT1, aplikacja musi upewnić się, że parametr *GroupId* ma odpowiednią wartość.

Grupy komunikatów i segmenty mogą być poprawnie przetwarzane tylko wtedy, gdy identyfikator grupy jest unikalny. Z tego powodu aplikacje *nie mogą generować własnych identyfikatorów grup* ; Zamiast tego aplikacje muszą wykonać jedną z następujących czynności:

- Jeśli określono parametr MQPMO_LOGICAL_ORDER, menedżer kolejek automatycznie generuje unikalny identyfikator grupy dla pierwszego komunikatu w grupie lub segmencie komunikatu logicznego i używa tego identyfikatora grupy dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego, dlatego aplikacja nie musi podejmować żadnych działań specjalnych. Jest to procedura zalecana.
- Jeśli parametr MQPMO_LOGICAL_ORDER nie jest określony, aplikacja musi zażądać od menedżera kolejek wygenerowania identyfikatora grupy, ustawiając wartość parametru *GroupId* na MQGI_NONE w pierwszym wywołaniu MQPUT lub MQPUT1 dla komunikatu w grupie lub segmencie komunikatu logicznego. Identyfikator grupy zwracany przez menedżera kolejek na wyjściu z tego wywołania musi być następnie używany dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego. Jeśli grupa komunikatów zawiera komunikaty posegmentowane, ten sam identyfikator grupy musi być używany dla wszystkich segmentów i komunikatów w grupie.

  Jeśli parametr MQPMO_LOGICAL_ORDER nie jest określony, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności (na przykład w kolejności odwrotnej), ale identyfikator grupy musi być przydzielany przez *pierwsze* wywołanie MQPUT lub MQPUT1 , które jest wysyłane dla dowolnego z tych komunikatów.

Na wejściu wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości opisanej w sekcji Kolejność fizyczna w kolejce. W danych wyjściowych wywołań MQPUT i MQPUT1 menedżer kolejek ustawia w tym polu wartość, która została wysłana z komunikatem, jeśli otwarty obiekt jest pojedynczą kolejką, a nie listą dystrybucyjną, ale pozostawia to pole bez zmian, jeśli otwarty obiekt jest listą dystrybucyjną. W tym drugim przypadku, jeśli aplikacja musi znać wygenerowane identyfikatory grup, musi udostępnić rekordy MQPMR zawierające pole *GroupId* .

Na wejściu wywołania MQGET menedżer kolejek używa wartości opisanej w sekcji Tabela 495 na stronie 403. W danych wyjściowych wywołania MQGET menedżer kolejek ustawia w tym polu wartość pobranego komunikatu.

Zdefiniowana jest następująca wartość specjalna:

MQGI_NONE
Nie określono identyfikatora grupy.

Wartością długości pola jest zero binarne. Jest to wartość używana dla komunikatów, które nie są w grupach, nie są segmentami komunikatów logicznych i dla których segmentacja nie jest dozwolona.

W języku programowania C zdefiniowana jest również stała MQGI_NONE_ARRAY. Ma ona taką samą wartość jak MQGI_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest określona przez wartość MQ_GROUP_ID_LENGTH. Wartością początkową tego pola jest MQGI_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

MsgSeqNumer (MQLONG) w deskryptorze MQMD

Jest to numer kolejny komunikatu logicznego w grupie.

Numery kolejne zaczynają się od 1 i zwiększają się o 1 dla każdego nowego komunikatu logicznego w grupie, maksymalnie do 999 999 999. Numerem kolejnym komunikatu fizycznego będącego poza grupą jest 1.

Aplikacja nie musi ustawiać tego pola w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono parametr MQPMO_LOGICAL_ORDER.
- W wywołaniu MQGET nie określono parametru MQMO_MATCH_MSG_SEQ_NUMBER.

Są to zalecane sposoby używania tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołanie ma wartość MQPUT1, aplikacja musi upewnić się, że parametr *MsgSeqNumber* ma odpowiednią wartość.

Na wejściu wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości opisanej w sekcji Kolejność fizyczna w kolejce. W danych wyjściowych wywołań MQPUT i MQPUT1 menedżer kolejek ustawia w tym polu wartość, która została wysłana razem z komunikatem.

Na wejściu wywołania MQGET menedżer kolejek używa wartości przedstawionej na rysunku (Tabela 495 na stronie 403). W danych wyjściowych wywołania MQGET menedżer kolejek ustawia w tym polu wartość pobranego komunikatu.

Wartością początkową tego pola jest jeden. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

Przesunięcie (MQLONG) dla deskryptora MQMD

Jest to przesunięcie w bajtach danych w komunikacie fizycznym od początku komunikatu logicznego, którego dane stanowią część. Te dane są nazywane *segmentem*. Przesunięcie jest z zakresu od 0 do 999 999 999. Komunikat fizyczny, który nie jest segmentem komunikatu logicznego, ma przesunięcie równe zero.

Aplikacja nie musi ustawiać tego pola w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono parametr MQPMO_LOGICAL_ORDER.
- W wywołaniu MQGET nie określono MQMO_MATCH_OFFSET.

Są to zalecane sposoby używania tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja nie spełnia tych warunków lub wywołanie ma wartość MQPUT1, aplikacja musi upewnić się, że parametr *Offset* ma odpowiednią wartość.

Na wejściu wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości opisanej w sekcji Kolejność fizyczna w kolejce. W danych wyjściowych wywołań MQPUT i MQPUT1 menedżer kolejek ustawia w tym polu wartość, która została wysłana razem z komunikatem.

W przypadku raportowania komunikatu raportu dotyczącego segmentu komunikatu logicznego pole *OriginalLength* (pod warunkiem, że nie jest to wartość MQOL_UNDEFINED) jest używane do aktualizowania przesunięcia w informacjach o segmencie przechowywanych przez menedżer kolejek.

Na wejściu wywołania MQGET menedżer kolejek używa wartości przedstawionej na rysunku (Tabela 495 na stronie 403). W danych wyjściowych wywołania MQGET menedżer kolejek ustawia w tym polu wartość pobranego komunikatu.

Wartością początkową tego pola jest zero. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

MsgFlags (MQLONG) w deskryptorze MQMD

MsgFlags to flagi, które określają atrybuty komunikatu lub sterują jego przetwarzaniem.

MsgFlags są podzielone na następujące kategorie:

- Flagi segmentacji
- Flagi statusu

Opcje segmentacji: Jeśli komunikat jest zbyt duży dla kolejki, próba umieszczenia komunikatu w kolejce zwykle kończy się niepowodzeniem. Segmentacja to technika, w której menedżer kolejek lub aplikacja dzieli komunikat na mniejsze części nazywane segmentami i umieszcza każdy segment w kolejce jako oddzielny komunikat fizyczny. Aplikacja pobierająca komunikat może pobrać segmenty jeden po drugim lub zażądać od menedżera kolejek ponownego złożenia segmentów w pojedynczy komunikat, który jest zwracany przez wywołanie MQGET. Ten ostatni jest osiągany przez określenie opcji MQGMO_COMPLETE_MSG w wywołaniu MQGET i podanie buforu, który jest wystarczająco duży, aby pomieścić cały komunikat. (Szczegółowe informacje na temat opcji MQGMO_COMPLETE_MSG zawiera sekcja "MQGMO-opcje pobierania komunikatów" na stronie 376). Komunikat może być segmentowany w nadawczym menedżerze kolejek, w pośrednim menedżerze kolejek lub w docelowym menedżerze kolejek.

Aby sterować segmentacją komunikatu, można określić jedną z następujących opcji:

MQMF_SEGMENTATION_INHIBITED
Ta opcja zapobiega rozbiciu komunikatu na segmenty przez menedżer kolejek. Jeśli ta opcja jest określona dla komunikatu, który jest już segmentem, zapobiega rozbiciu segmentu na mniejsze segmenty.

Wartością tej flagi jest zero binarne. Jest to opcja domyślna.

MQMF_SEGMENTATION_ALLOWED
Ta opcja umożliwia podzieleniem komunikatu na segmenty przez menedżer kolejek. Jeśli ta opcja jest określona dla komunikatu, który jest już segmentem, umożliwia podział segmentu na mniejsze segmenty. Parametr MQMF_SEGMENTATION_ALLOWED można ustawić bez ustawiania parametru MQMF_SEGMENT lub MQMF_LAST_SEGMENT.

- W systemie z/OSmenedżer kolejek nie obsługuje segmentacji komunikatów. Jeśli komunikat jest zbyt duży dla kolejki, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_MSG_TOO_BIG_FOR_Q. Jednak opcja MQMF_SEGMENTATION_ALLOWED może być nadal określona i umożliwia segmentację komunikatu w zdalnym menedżerze kolejek.

Gdy menedżer kolejek segmentuje komunikat, menedżer kolejek włącza flagę MQMF_SEGMENT w kopii deskryptora MQMD wysyłanej z każdym segmentem, ale nie zmienia ustawień tych flag w deskryptorze MQMD udostępnianym przez aplikację w wywołaniu MQPUT lub MQPUT1 . Dla ostatniego segmentu w komunikacie logicznym menedżer kolejek również włącza flagę MQMF_LAST_SEGMENT w deskryptorze MQMD wysyłanym razem z tym segmentem.

Uwaga: Należy zachować ostrożność podczas umieszczania komunikatów z opcją MQMF_SEGMENTATION_ALLOWED, ale bez opcji MQPMO_LOGICAL_ORDER. Jeśli komunikat jest następujący:

- Nie jest segmentem, oraz
- Nie należy do grupy, oraz
- Nie jest przekazywane,

Aplikacja musi zresetować pole *GroupId* na wartość MQGI_NONE przed *każdym wywołaniem* MQPUT lub MQPUT1 , aby menedżer kolejek mógł wygenerować unikalny identyfikator grupy dla każdego komunikatu. Jeśli ta czynność nie zostanie wykonana, niepowiązane komunikaty mogą mieć ten sam identyfikator grupy, co może prowadzić do późniejszego niepoprawnego przetwarzania. Więcej informacji na temat resetowania pola *GroupId* zawierają opisy pól *GroupId* i MQPMO_LOGICAL_ORDER.

W razie potrzeby menedżer kolejek dzieli komunikaty na segmenty, aby segmenty (wraz z wymaganymi danymi nagłówka) zmieściły się w kolejce. Istnieje jednak dolny limit wielkości segmentu wygenerowanego przez menedżer kolejek i tylko ostatni segment utworzony na podstawie komunikatu może być mniejszy niż ten limit (dolny limit wielkości segmentu wygenerowanego przez aplikację wynosi jeden bajt). Segmenty wygenerowane przez menedżera kolejek mogą mieć nierówną długość. Menedżer kolejek przetwarza komunikat w następujący sposób:

- Formaty zdefiniowane przez użytkownika są dzielone na granice, które są wielokrotnością 16 bajtów; menedżer kolejek nie generuje segmentów mniejszych niż 16 bajtów (innych niż ostatni segment).
- Wbudowane formaty inne niż MQFMT_STRING są dzielone w punktach odpowiednich do rodzaju obecnych danych. Jednak menedżer kolejek nigdy nie dzieli komunikatu w środku struktury nagłówka IBM MQ . Oznacza to, że segment zawierający pojedynczą strukturę nagłówka produktu MQ nie może zostać dalej podzielony przez menedżer kolejek, w związku z czym minimalna możliwa wielkość segmentu dla tego komunikatu jest większa niż 16 bajtów.

  Drugi lub późniejszy segment wygenerowany przez menedżer kolejek rozpoczyna się od jednego z następujących członów:

  – Struktura nagłówka MQ
  – Początek danych komunikatu aplikacji
  – Część drogi przez dane komunikatu aplikacji
- Parametr MQFMT_STRING jest dzielony bez względu na rodzaj obecnych danych (SBCS, DBCS lub mieszany SBCS/DBCS). Jeśli łańcuch jest typu DBCS lub mieszanego SBCS/DBCS, może to spowodować, że segmenty nie będą mogły zostać przekształcone z jednego zestawu znaków na inny. Menedżer kolejek nigdy nie dzieli komunikatów MQFMT_STRING na segmenty mniejsze niż 16 bajtów (inne niż ostatni segment).
- Menedżer kolejek ustawia pola *Format*, *CodedCharSetId* i *Encoding* w deskryptorze MQMD każdego segmentu w celu poprawnego opisania danych obecnych na *początku* segmentu. Nazwa formatu jest nazwą wbudowanego formatu lub nazwą formatu zdefiniowanego przez użytkownika.
- Modyfikowane jest pole *Report* w strukturze MQMD segmentów z wartością *Offset* większą niż zero. Dla każdego typu raportu, jeśli opcja raportu to MQRO_ * _WITH_DATA, ale segment nie może zawierać żadnego z pierwszych 100 bajtów danych użytkownika (czyli danych następujących po dowolnych strukturach nagłówka IBM MQ , które mogą być obecne), opcja raportu zostanie zmieniona na MQRO_ *.

Menedżer kolejek postępuje zgodnie z powyższymi regułami, ale w przeciwnym razie dzieli komunikaty nieprzewidywalnie. Nie należy zakładać, gdzie komunikat jest dzielony.

W przypadku *trwałych* komunikatów menedżer kolejek może wykonać segmentację tylko w obrębie jednostki pracy:

- Jeśli wywołanie MQPUT lub MQPUT1 działa w obrębie jednostki pracy zdefiniowanej przez użytkownika, używana jest ta jednostka pracy. Jeśli wywołanie nie powiedzie się podczas procesu segmentacji, menedżer kolejek usuwa wszystkie segmenty, które zostały umieszczone w kolejce w wyniku wywołania zakończonego niepowodzeniem. Jednak niepowodzenie nie uniemożliwia pomyślnego zatwierdzenia jednostki pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika i nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy tylko na czas trwania wywołania. Jeśli wywołanie powiedzie się, menedżer kolejek automatycznie zatwierdza jednostkę pracy. Jeśli wywołanie nie powiedzie się, menedżer kolejek wycofa jednostkę pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika, ale istnieje jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek nie może wykonać segmentacji. Jeśli komunikat nie wymaga segmentacji, wywołanie nadal może zakończyć się powodzeniem. Jeśli jednak komunikat wymaga segmentacji, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_UOW_NOT_AVAILABLE.

W przypadku komunikatów *nietrwałych* menedżer kolejek nie wymaga, aby jednostka pracy była dostępna na potrzeby segmentacji.

Należy zachować szczególną ostrożność podczas przekształcania danych w komunikatach, które mogą być segmentowane:

- Jeśli aplikacja odbierająca przekształca dane w wywołaniu MQGET i określa opcję MQGMO_COMPLETE_MSG, wyjście konwersji danych jest przekazywane do kompletnego komunikatu dla wyjścia do przekształcenia, a fakt segmentacji komunikatu jest widoczny dla wyjścia.
- Jeśli aplikacja odbierająca pobiera jeden segment na raz, wywoływane jest wyjście konwersji danych w celu przekształcenia jednego segmentu na raz. W związku z tym wyjście musi przekształcić dane w segmencie niezależnie od danych w dowolnym innym segmencie.

  Jeśli rodzaj danych w komunikacie jest taki, że dowolna segmentacja danych w granicach 16-bajtowych może spowodować, że segmenty nie będą mogły być przekształcane przez wyjście lub że format to MQFMT_STRING, a zestaw znaków to DBCS lub mieszane SBCS/DBCS, aplikacja wysyłająca musi utworzyć i umieścić segmenty, określając parametr MQMF_SEGMENTATION_INHIBITED w celu pominięcia dalszej segmentacji. W ten sposób aplikacja wysyłająca może zapewnić, że każdy segment zawiera informacje wystarczające do pomyślnego przekształcenia segmentu przez wyjście konwersji danych.
- Jeśli konwersja nadawcy jest określona dla agenta kanału komunikatów wysyłania (MCA), agent MCA przekształca tylko te komunikaty, które nie są segmentami komunikatów logicznych. Agent MCA nigdy nie podejmuje próby przekształcenia komunikatów, które są segmentami.

Ta opcja jest flagą wejściową wywołań MQPUT i MQPUT1 oraz flagą wyjściową wywołania MQGET. W tym drugim wywołaniu menedżer kolejek również echo wartości flagi w polu *Segmentation* w MQGMO.

Wartością początkową tej opcji jest MQMF_SEGMENTATION_INHIBITED.

Flagi statusu: Są to flagi wskazujące, czy komunikat fizyczny należy do grupy komunikatów, czy jest segmentem komunikatu logicznego, czy też nie. W wywołaniu MQPUT lub MQPUT1 albo zwracanym przez wywołanie MQGET można określić co najmniej jeden z następujących elementów:

MQMF_MSG_IN_GROUP
Komunikat jest elementem grupy.

MQMF_LAST_MSG_IN_GROUP
Komunikat jest ostatnim logicznym komunikatem w grupie.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza opcję MQMF_MSG_IN_GROUP w kopii deskryptora MQMD wysyłanej z komunikatem, ale nie zmienia ustawień tych opcji w deskryptorze MQMD udostępnianym przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Poprawne jest, aby grupa składała się tylko z jednego komunikatu logicznego. W takim przypadku parametr MQMF_LAST_MSG_IN_GROUP jest ustawiony, ale pole *MsgSeqNumber* ma wartość 1.

MQMF_SEGMENT
Komunikat jest segmentem komunikatu logicznego.

Jeśli parametr MQMF_SEGMENT jest określony bez parametru MQMF_LAST_SEGMENT, długość danych komunikatu aplikacji w segmencie (*wykluczając* długość ewentualnych struktur nagłówka IBM MQ) musi wynosić co najmniej jeden. Jeśli długość wynosi zero, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_SEGMENT_LENGTH_ZERO.

W systemie z/OSta opcja nie jest obsługiwana, jeśli komunikat jest umieszczany w kolejce o typie indeksu MQIT_GROUP_ID.

MQMF_LAST_SEGMENT
Komunikat jest ostatnim segmentem komunikatu logicznego.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza opcję MQMF_SEGMENT w kopii deskryptora MQMD wysyłanej z komunikatem, ale nie zmienia ustawień tych opcji w deskryptorze MQMD udostępnianym przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Komunikat logiczny może składać się tylko z jednego segmentu. Jeśli tak, to pole MQMF_LAST_SEGMENT jest ustawione, ale pole *Offset* ma wartość zero.

Jeśli określono opcję MQMF_LAST_SEGMENT, długość danych komunikatu aplikacji w segmencie (*wykluczając* długość dowolnej struktury nagłówka, która może być obecna) może wynosić zero.

W systemie z/OSta opcja nie jest obsługiwana, jeśli komunikat jest umieszczany w kolejce o typie indeksu MQIT_GROUP_ID.

Aplikacja musi upewnić się, że te flagi są poprawnie ustawione podczas umieszczania komunikatów. Jeśli określono parametr MQPMO_LOGICAL_ORDER lub określono go w poprzedzającym wywołaniu MQPUT dla uchwytu kolejki, ustawienia flag muszą być spójne z informacjami o grupie i segmencie zachowanymi przez menedżer kolejek dla uchwytu kolejki. Poniższe warunki mają zastosowanie do *kolejnych* wywołań MQPUT dla uchwytu kolejki, jeśli określono parametr MQPMO_LOGICAL_ORDER:

- Jeśli nie ma bieżącej grupy lub komunikatu logicznego, wszystkie te flagi (i ich kombinacje) są poprawne.
- Po określeniu parametru MQMF_MSG_IN_GROUP musi on pozostać włączony do czasu określenia parametru MQMF_LAST_MSG_IN_GROUP. Wywołanie nie powiodło się z kodem przyczyny MQRC_INCOMPLETE_GROUP, jeśli ten warunek nie został spełniony.
- Po określeniu parametru MQMF_SEGMENT musi on pozostać włączony do czasu określenia parametru MQMF_LAST_SEGMENT. Wywołanie nie powiedzie się z kodem przyczyny MQRC_INCOMPLETE_MSG, jeśli ten warunek nie jest spełniony.
- Jeśli parametr MQMF_SEGMENT został określony bez parametru MQMF_MSG_IN_GROUP, parametr MQMF_MSG_IN_GROUP musi pozostać *wyłączony* , dopóki nie zostanie określony parametr MQMF_LAST_SEGMENT. Wywołanie nie powiedzie się z kodem przyczyny MQRC_INCOMPLETE_MSG, jeśli ten warunek nie jest spełniony.

Kolejność fizyczna w kolejce wyświetla poprawne kombinacje flag i wartości używane w różnych polach.

Są to flagi wejściowe w wywołaniach MQPUT i MQPUT1 oraz flagi wyjściowe w wywołaniu MQGET. W drugim wywołaniu menedżer kolejek również echa wartości flag do pól *GroupStatus* i *SegmentStatus* w MQGMO.

W przypadku publikowania/subskrybowania nie można używać zgrupowanych ani posegmentowanych komunikatów.

Opcje domyślne: Aby wskazać, że komunikat ma atrybuty domyślne, można określić następujące wartości:

MQMF_BRAK
Brak flag komunikatów (domyślne atrybuty komunikatów).

Powoduje to zablokowanie segmentacji i wskazuje, że komunikat nie znajduje się w grupie i nie jest segmentem komunikatu logicznego. Parametr MQMF_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta flaga była używana z innymi, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Pole *MsgFlags* jest podzielone na podpola; szczegółowe informacje zawiera sekcja “Opcje raportu i flagi komunikatów” na stronie 933.

Wartością początkową tego pola jest MQMF_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

OriginalLength (MQLONG) dla deskryptora MQMD

To pole dotyczy tylko komunikatów raportu, które są segmentami. Określa ona długość segmentu komunikatu, do którego odnosi się komunikat raportu; nie określa długości komunikatu logicznego, którego część stanowi segment, ani długości danych w komunikacie raportu.

Uwaga: Podczas generowania komunikatu raportu dla komunikatu, który jest segmentem, menedżer kolejek i agent kanału komunikatów kopiują do deskryptora MQMD dla komunikatu raportu pola *GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags*z oryginalnego komunikatu. W rezultacie komunikat raportu jest

również segmentem. Aplikacje generujące komunikaty raportu muszą działać tak samo i poprawnie ustawić pole *OriginalLength* .

Zdefiniowana jest następująca wartość specjalna:

MQOL_UNDEFINED
Pierwotna długość komunikatu nie została zdefiniowana.

OriginalLength to pole wejściowe w wywołaniach MQPUT i MQPUT1 , ale wartość udostępniana przez aplikację jest akceptowana tylko w określonych okolicznościach:

- Jeśli umieszczany komunikat jest segmentem, a także komunikatem raportu, menedżer kolejek akceptuje podaną wartość. Wartość musi być następująca:
  - Wartość większa od zera, jeśli segment nie jest ostatnim segmentem
  - Nie mniej niż zero, jeśli segment jest ostatnim segmentem
  - Nie mniej niż długość danych obecnych w komunikacie

  Jeśli te warunki nie są spełnione, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ORIGINAL_LENGTH_ERROR.
- Jeśli umieszczany komunikat jest segmentem, ale nie komunikatem raportu, menedżer kolejek ignoruje pole i zamiast niego używa długości danych komunikatu aplikacji.
- We wszystkich innych przypadkach menedżer kolejek ignoruje to pole i używa zamiast niego wartości MQOL_UNDEFINED.

Jest to pole wyjściowe wywołania MQGET.

Wartością początkową tego pola jest MQOL_UNDEFINED. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQMD_VERSION_2.

MQMDE-rozszerzenie deskryptora komunikatu

Struktura MQMDE opisuje dane, które czasami występują przed danymi komunikatu aplikacji. Struktura zawiera te pola MQMD, które istnieją w version-2 MQMD, ale nie w version-1 MQMD.

Dostępność

Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Nazwa formatu

MQFMT_MD_EXTENSION

Zestaw znaków i kodowanie

Dane w MQMDE muszą być w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one udostępniane przez atrybut menedżera kolejek systemu **CodedCharSetId** i parametr MQENC_NATIVE dla języka programowania C.

Ustaw zestaw znaków i kodowanie MQMDE w polach *CodedCharSetId* i *Encoding* w następujących polach:

- MQMD (jeśli struktura MQMDE jest na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQMDE (wszystkie inne przypadki).

Jeśli produkt MQMDE nie znajduje się w zestawie znaków i kodowaniu menedżera kolejek, produkt MQMDE jest akceptowany, ale nie jest honorowany, co oznacza, że produkt MQMDE jest traktowany jako dane komunikatu.

Uwaga: W systemie Windowsaplikacje skompilowane z programem Micro Focus COBOL używają wartości MQENC_NATIVE innej niż kodowanie menedżera kolejek. Chociaż pola liczbowe w strukturze MQMD wywołań MQPUT, MQPUT1i MQGET muszą być zakodowane w języku Micro Focus COBOL, pola liczbowe

w strukturze MQMDE muszą być zakodowane w menedżerze kolejek. Ta druga wartość jest podawana przez funkcję MQENC_NATIVE dla języka programowania C i ma wartość 546.

Użycie

Aplikacje używające deskryptora MQMD version-2 , nie napotkają struktury MQMDE. Jednak w niektórych sytuacjach wyspecjalizowane aplikacje i aplikacje, które nadal używają deskryptora MQMD version-1 , mogą napotkać interfejs MQMDE. Struktura MQMDE może wystąpić w następujących okolicznościach:

- Określone w wywołaniach MQPUT i MQPUT1
- Zwracane przez wywołanie MQGET
- W komunikatach w kolejkach transmisji

MQMDE określone w wywołaniach MQPUT i MQPUT1

W wywołaniach MQPUT i MQPUT1 , jeśli aplikacja udostępnia deskryptor MQMD version-1 , aplikacja może opcjonalnie poprzedzić dane komunikatu przedrostkiem MQMDE, ustawiając pole *Format* w deskryptorze MQMD na wartość MQFMT_MD_EXTENSION w celu wskazania, że istnieje MQMDE. Jeśli aplikacja nie udostępnia środowiska MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w środowisku MQMDE. Wartości domyślne używane przez menedżer kolejek są takie same jak wartości początkowe struktury; patrz sekcja Tabela 503 na stronie 487.

Jeśli aplikacja udostępnia version-2 MQMD *i* poprzedza dane komunikatu aplikacji przedrostkiem MQMDE, struktury są przetwarzane w sposób przedstawiony w poniższej tabeli.

Tabela 502. Działanie menedżera kolejek, gdy określono MQMDE w MQPUT lub MQPUT1 dla MQMDE

Wersja MQMD	Wartości pól version-2	Wartości odpowiednich pól w MQMDE	Działanie wykonywane przez menedżer kolejek
1	-	Ważne	MQMDE jest honorowane
2	Domyślny	Ważne	MQMDE jest honorowane
2	Niedomyślna	Ważne	MQMDE jest traktowane jako dane komunikatu
1 lub 2	Dowolna	Niepoprawne	Wywołanie nie powiodło się z odpowiednim kodem przyczyny
1 lub 2	Dowolna	MQMDE jest w niepoprawnym zestawie znaków lub kodowaniu albo jest nieobsługiwaną wersją	MQMDE jest traktowane jako dane komunikatu

Uwaga: W systemie z/OS, jeśli aplikacja określa strukturę MQMD version-1 z MQMDE, menedżer kolejek sprawdza poprawność struktury MQMDE tylko wtedy, gdy kolejka ma identyfikator *IndexType* o wartości MQIT_GROUP_ID.

Jest jeden szczególny przypadek. Jeśli aplikacja używa deskryptora MQMD version-2 w celu umieszczenia komunikatu, który jest segmentem (flaga MQMF_SEGMENT lub MQMF_LAST_SEGMENT jest ustawiona), a nazwa formatu w deskryptorze MQMD to MQFMT_DEAD_LETTER_HEADER, menedżer kolejek generuje strukturę MQMDE i wstawia ją *między* strukturą MQDLH i danymi znajdującymi się po niej. W strukturze MQMD, w której menedżer kolejek zachowuje komunikat, pola version-2 są ustawione na wartości domyślne.

Niektóre z pól, które istnieją w programie MQMD version-2 , ale nie w programie MQMD version-1 , są polami wejściowymi/wyjściowymi MQPUT i MQPUT1. Jednak menedżer kolejek nie zwraca żadnych wartości w odpowiednich polach w MQMDE na wyjściu z wywołań MQPUT i MQPUT1 . Jeśli aplikacja wymaga tych wartości wyjściowych, musi użyć deskryptora MQMD version-2 .

MQMDE zwrócone przez wywołanie MQGET

W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD version-1 , menedżer kolejek dodaje przedrostek do komunikatu zwróconego z MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość inną niż domyślna. Menedżer kolejek ustawia pole *Format* w strukturze MQMD na wartość MQFMT_MD_EXTENSION, aby wskazać, że istnieje struktura MQMDE.

Jeśli aplikacja udostępnia środowisko MQMDE na początku parametru **Buffer** , środowisko MQMDE jest ignorowane. Po powrocie z wywołania MQGET jest on zastępowany przez MQMDE dla komunikatu (jeśli jest potrzebny) lub nadpisywany przez dane komunikatu aplikacji (jeśli nie jest potrzebny MQMDE).

Jeśli wywołanie MQGET zwraca MQMDE, dane w MQMDE są zwykle w zestawie znaków i kodowaniu menedżera kolejek. Jednak MQMDE może mieć inny zestaw znaków i kodowanie, jeśli:

- Środowisko MQMDE było traktowane jako dane w wywołaniu MQPUT lub MQPUT1 (informacje na temat okoliczności, które mogą spowodować tę sytuację, zawiera sekcja Tabela 502 na stronie 486).
- Komunikat został odebrany od menedżera kolejek zdalnych połączonego połączeniem TCP, a agent kanału komunikatów odbierających (MCA) nie został poprawnie skonfigurowany.

Uwaga: W systemie Windowsaplikacje skompilowane przy użyciu języka Micro Focus COBOL używają wartości MQENC_NATIVE innej niż kodowanie menedżera kolejek (patrz wyżej).

MQMDE w komunikatach w kolejkach transmisji

Komunikaty w kolejkach transmisji są poprzedzone strukturą MQXQH, która zawiera w sobie strukturę MQMD version-1 . Produkt MQMDE może być również obecny, umieszczony między strukturą MQXQH i danymi komunikatu aplikacji, ale zwykle występuje tylko wtedy, gdy co najmniej jedno pole w produkcie MQMDE ma wartość inną niż domyślna.

Inne struktury nagłówka produktu MQ mogą również występować między strukturą MQXQH a danymi komunikatu aplikacji. Jeśli na przykład istnieje nagłówek niedostarczonego komunikatu MQDLH, a komunikat nie jest segmentem, kolejność jest następująca:

- MQXQH (zawierający version-1 MQMD)
- MQMDE,
- MQDLH
- dane komunikatu aplikacji

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 503. Pola w MQMDE dla MQMDE

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQMDE_STRUC_ID (Identyfikator struktury menedżera kolejek)	'MDE¬'
Wersja (numer wersji struktury)	MQMDE_VERSION_2	2
StrucLength (długość struktury MQMDE)	MQMDE_LENGTH_2	72
Kodowanie (kodowanie liczbowe danych następujących po MQMDE)	RODZIMA MQENC	Zależy od środowiska
CodedCharSetId (identyfikator zestawu znaków danych następujący po MQMDE)	MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0

Tabela 503. Pola w MQMDE dla MQMDE (kontynuacja)

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
Format (nazwa formatu danych po MQMDE)	MQFMT_BRAK	Puste
Flagi (flagi ogólne)	MQMDEF_BRAK	0
GroupId (identyfikator grupy)	MQGI_NONE	Wartości null
MsgSeqNumber (numer kolejny komunikatu logicznego w grupie)	Brak	1
Przesunięcie (przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego)	Brak	0
MsgFlags (flaga komunikatu)	MQMF_BRAK	0
OriginalLength (długość oryginalnej wiadomości)	MQOL_UNDEFINED	-1

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraZmienna MQMDE_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQMDE MyMDE = {MQMDE_DEFAULT};
```

Deklaracje językowe

Deklaracja języka C dla MQMDE

```
typedef struct tagMQMDE MQMDE;
struct tagMQMDE {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQMDE structure */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQMDE */
  MQLONG    CodedCharSetId;  /* Character-set identifier of data that
                                follows MQMDE */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQMDE */
  MQLONG    Flags;           /* General flags */
  MQBYTE24  GroupId;         /* Group identifier */
  MQLONG    MsgSeqNumber;    /* Sequence number of logical message
                                within group */
  MQLONG    Offset;          /* Offset of data in physical message from
                                start of logical message */
  MQLONG    MsgFlags;        /* Message flags */
  MQLONG    OriginalLength;  /* Length of original message */
};
```

Deklaracja języka COBOL dla MQMDE

```
**   MQMDE structure
  10 MQMDE.
**    Structure identifier
   15 MQMDE-STRUCID        PIC X(4).
**    Structure version number
   15 MQMDE-VERSION        PIC S9(9) BINARY.
**    Length of MQMDE structure
   15 MQMDE-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows MQMDE
   15 MQMDE-ENCODING       PIC S9(9) BINARY.
**    Character-set identifier of data that follows MQMDE
   15 MQMDE-CODEDCHARSETID PIC S9(9) BINARY.
```

```
**    Format name of data that follows MQMDE
   15 MQMDE-FORMAT         PIC X(8).
**    General flags
   15 MQMDE-FLAGS          PIC S9(9) BINARY.
**    Group identifier
   15 MQMDE-GROUPID        PIC X(24).
**    Sequence number of logical message within group
   15 MQMDE-MSGSEQNUMBER   PIC S9(9) BINARY.
**    Offset of data in physical message from start of logical message
   15 MQMDE-OFFSET         PIC S9(9) BINARY.
**    Message flags
   15 MQMDE-MSGFLAGS       PIC S9(9) BINARY.
**    Length of original message
   15 MQMDE-ORIGINALLENGTH PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQMDE

```
dcl
 1 MQMDE based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Length of MQMDE structure */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQMDE */
  3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                     that follows MQMDE */
  3 Format         char(8),       /* Format name of data that follows
                                     MQMDE */
  3 Flags          fixed bin(31), /* General flags */
  3 GroupId        char(24),      /* Group identifier */
  3 MsgSeqNumber   fixed bin(31), /* Sequence number of logical message
                                     within group */
  3 Offset         fixed bin(31), /* Offset of data in physical message
                                     from start of logical message */
  3 MsgFlags       fixed bin(31), /* Message flags */
  3 OriginalLength fixed bin(31); /* Length of original message */
```

Deklaracja High Level Assembler dla MQMDE

```
MQMDE                 DSECT
MQMDE_STRUCID         DS   CL4   Structure identifier
MQMDE_VERSION         DS   F     Structure version number
MQMDE_STRUCLENGTH     DS   F     Length of MQMDE structure
MQMDE_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQMDE
MQMDE_CODEDCHARSETID  DS   F     Character-set identifier of data that
*                                follows MQMDE
MQMDE_FORMAT          DS   CL8   Format name of data that follows MQMDE
MQMDE_FLAGS           DS   F     General flags
MQMDE_GROUPID         DS   XL24  Group identifier
MQMDE_MSGSEQNUMBER    DS   F     Sequence number of logical message
*                                within group
MQMDE_OFFSET          DS   F     Offset of data in physical message from
*                                start of logical message
MQMDE_MSGFLAGS        DS   F     Message flags
MQMDE_ORIGINALLENGTH  DS   F     Length of original message
*
MQMDE_LENGTH          EQU  *-MQMDE
                      ORG  MQMDE
MQMDE_AREA            DS   CL(MQMDE_LENGTH)
```

Deklaracja Visual Basic dla MQMDE

```
Type MQMDE
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Length of MQMDE structure'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'MQMDE'
  CodedCharSetId As Long     'Character-set identifier of data that'
                             'follows MQMDE'
  Format         As String*8 'Format name of data that follows MQMDE'
  Flags          As Long     'General flags'
  GroupId        As MQBYTE24 'Group identifier'
  MsgSeqNumber   As Long     'Sequence number of logical message within'
```

```
                              'group'
  Offset         As Long      'Offset of data in physical message from'
                              'start of logical message'
  MsgFlags       As Long      'Message flags'
  OriginalLength As Long      'Length of original message'
End Type
```

StrucId (MQCHAR4) dla MQMDE

Jest to identyfikator struktury struktury rozszerzenia deskryptora komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQMDE_STRUC_ID.

Wartość musi być następująca:

MQMDE_STRUC_ID (Identyfikator struktury menedżera kolejek)
Identyfikator struktury rozszerzenia deskryptora komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQMDE_STRUC_ID_ARRAY. Ma taką samą wartość jak zmienna MQMDE_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQMDE

Jest to numer wersji struktury; wartość musi być następująca:

MQMDE_VERSION_2
Struktura rozszerzenia deskryptora komunikatu Version-2 .

Następująca stała określa numer wersji bieżącej:

MQMDE_CURRENT_VERSION
Bieżąca wersja struktury rozszerzenia deskryptora komunikatu.

Wartością początkową tego pola jest MQMDE_VERSION_2.

StrucLength (MQLONG) dla MQMDE

Jest to długość struktury MQMDE. Zdefiniowana jest następująca wartość:

MQMDE_LENGTH_2
Długość struktury rozszerzenia deskryptora komunikatu version-2 .

Wartością początkową tego pola jest MQMDE_LENGTH_2.

Kodowanie (MQLONG) dla MQMDE

Określa kodowanie liczbowe danych, które są zgodne ze strukturą MQMDE. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Więcej informacji na temat kodowania danych zawiera opis pola *Encoding* w sekcji “MQMD-deskryptor komunikatu” na stronie 432 .

Wartością początkową tego pola jest MQENC_NATIVE.

CodedCharSetId (MQLONG) dla MQMDE

Określa identyfikator zestawu znaków danych, które są zgodne ze strukturą MQMDE. Nie ma on zastosowania do danych znakowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Pola MQCCSI_INHERIT nie można używać, jeśli wartością pola *PutApplType* w deskryptorze MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQMDE

Określa nazwę formatu danych, które są zgodne ze strukturą MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Więcej informacji na temat nazw formatów zawiera opis pola *Format* w sekcji"MQMD-deskryptor komunikatu" na stronie 432.

Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQMDE

Można podać następującą opcję:

MQMDEF_BRAK
Brak flag.

Wartością początkową tego pola jest MQMDEF_NONE.

GroupId (MQBYTE24) dla MQMDE

Patrz opis pola *GroupId* w sekcji"MQMD-deskryptor komunikatu" na stronie 432. Wartością początkową tego pola jest MQGI_NONE.

MsgSeqNumer (MQLONG) w MQMDE

Patrz opis pola *MsgSeqNumber* w sekcji"MQMD-deskryptor komunikatu" na stronie 432. Wartością początkową tego pola jest 1.

Przesunięcie (MQLONG) dla MQMDE

Patrz opis pola *Offset* w sekcji"MQMD-deskryptor komunikatu" na stronie 432. Wartością początkową tego pola jest 0.

MsgFlags (MQLONG) w środowisku MQMDE

Patrz opis pola *MsgFlags* w sekcji "MQMD-deskryptor komunikatu" na stronie 432. Wartością początkową tego pola jest MQMF_NONE.

OriginalLength (MQLONG) dla MQMDE

Patrz opis pola *OriginalLength* w sekcji"MQMD-deskryptor komunikatu" na stronie 432. Wartością początkową tego pola jest MQOL_UNDEFINED.

MQMHBO-opcje przesyłania komunikatu do buforu

Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia buforów z uchwytów komunikatów. Struktura jest parametrem wejściowym wywołania MQMHBUF.

Zestaw znaków i kodowanie

Dane w obiekcie MQMHBO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (MQENC_NATIVE).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 504. Pola w MQMHBO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQMHBO_STRUC_ID (Identyfikator struktury menedżera kolejek)	'MHBO'
Wersja (numer wersji struktury)	MQMHBO_VERSION_1	1
Opcje (opcje sterujące działaniem MQMHBUF)	MQMHBO_PROPERTIES_IN_MQRFH2	

Uwagi:

1. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
2. W języku programowania C: zmienna makraMQMHBO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQMHBO MyMHBO = {MQMHBO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQMHBO

```
typedef struct tagMQMHBO MQMHBO;
struct tagMQMHBO {
  MQCHAR4  StrucId;        /* Structure identifier */
  MQLONG   Version;        /* Structure version number */
  MQLONG   Options;        /* Options that control the action of
                              MQMHBUF */
};
```

Deklaracja języka COBOL dla MQMHBO

```
**  MQMHBO structure
  10 MQMHBO.
**    Structure identifier
   15 MQMHBO-STRUCID               PIC X(4).
**    Structure version number
   15 MQMHBO-VERSION               PIC S9(9) BINARY.
**    Options that control the action of MQMHBUF
   15 MQMHBO-OPTIONS               PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQMHBO

```
Dcl
 1 MQMHBO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
```

```
  3 Options        fixed bin(31), /* Options that control the action
                                     of MQMHBUF */
```

Deklaracja High Level Assembler dla MQMHBO

```
MQMHBO                      DSECT
MQMHBO_STRUCID              DS   CL4  Structure identifier
MQMHBO_VERSION              DS   F    Structure version number
MQMHBO_OPTIONS              DS   F    Options that control the
*                                     action of MQMHBUF
MQMHBO_LENGTH               EQU  *-MQMHBO
MQMHBO_AREA                 DS   CL(MQMHBO_LENGTH)
```

StrucId (MQCHAR4) dla MQMHBO

Jest to identyfikator struktury uchwytu komunikatu do struktury opcji buforu. Jest to zawsze pole wejściowe. Jego wartością jest MQMHBO_STRUC_ID.

Wartość musi być następująca:

MQMHBO_STRUC_ID (Identyfikator struktury menedżera kolejek)
Identyfikator struktury opcji buforu dla uchwytu komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQMHBO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQMHBO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQMHBO

Struktura opcji uchwytu komunikatu do buforu-pole Wersja

Jest to numer wersji struktury. Wartość musi być następująca:

MQMHBO_VERSION_1
Numer wersji struktury opcji przesyłania komunikatu do buforu.

Następująca stała określa numer wersji bieżącej:

MQMHBO_CURRENT_VERSION
Bieżąca wersja struktury opcji przesyłania komunikatów do buforu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQMHBO_VERSION_1.

Opcje (MQLONG) dla MQMHBO

Struktura z uchwytem komunikatu do opcji buforu-pole Opcje

Te opcje sterują działaniem MQMHBUF.

Należy podać następującą opcję:

MQMHBO_PROPERTIES_IN_MQRFH2
Podczas przekształcania właściwości z uchwytu komunikatu w bufor należy przekształcić je w format MQRFH2 .

Opcjonalnie można również określić następującą opcję. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

MQMHBO_DELETE_PROPERTIES
Właściwości dodawane do buforu są usuwane z uchwytu komunikatu. Jeśli wywołanie nie powiedzie się, nie zostaną usunięte żadne właściwości.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQMHBO_PROPERTIES_IN_MQRFH2.

MQOD-deskryptor obiektu

Struktura MQOD służy do określania obiektu według nazwy. Struktura jest parametrem wejścia/wyjścia w wywołaniach MQOPEN i MQPUT1 .

Poprawne są następujące typy obiektów:

- Kolejka lub lista dystrybucyjna
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- Temat

Dostępność

Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Wersja

Bieżąca wersja programu MQOD to MQOD_VERSION_4. Aplikacje, które mają być obsługiwane między kilkoma środowiskami, muszą mieć pewność, że wymagana wersja MQOD jest obsługiwana we wszystkich odnośnych środowiskach. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję produktu MQOD, która jest obsługiwana przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQOD_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *Version* numer wersji wymaganej wersji.

Aby można było otworzyć listę dystrybucyjną, *Version* musi mieć wartość MQOD_VERSION_2 lub większą.

Zestaw znaków i kodowanie

Dane w programie MQOD muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQOD_STRUC_ID (identyfikator struktury MQD)	'OD¬¬'
Wersja (numer wersji struktury)	MQOD_VERSION_1	1
ObjectType (typ obiektu)	MQOT_Q	1
ObjectName (nazwa obiektu)	Brak	Pusty łańcuch lub odstępy
ObjectQMgrNazwa (nazwa menedżera kolejek obiektu)	Brak	Pusty łańcuch lub odstępy
DynamicQName (nazwa kolejki dynamicznej)	Brak	'CSQ.*' na platformie z/OS ; 'AMQ.*' w przeciwnym razie

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
AlternateUserId (alternatywny identyfikator użytkownika)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_2.		
RecsPresent (liczba rekordów obiektów)	Brak	0
KnownDestCount (liczba pomyślnie otwartych kolejek lokalnych)	Brak	0
UnknownDestLiczba (liczba pomyślnie otwartych kolejek zdalnych)	Brak	0
InvalidDestLiczba (liczba kolejek, których otwarcie nie powiodło się)	Brak	0
ObjectRecPrzesunięcie (przesunięcie pierwszego rekordu obiektu od początku MQOD)	Brak	0
ResponseRec (przesunięcie pierwszego rekordu odpowiedzi od początku MQOD)	Brak	0
ObjectRecPtr (adres pierwszego rekordu obiektu)	Brak	Pusty wskaźnik lub puste bajty
ResponseRecPtr (adres pierwszego rekordu odpowiedzi)	Brak	Pusty wskaźnik lub puste bajty
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_3.		
AlternateSecurityId (alternatywny identyfikator zabezpieczeń)	MQSID_BRAK	Wartości null
ResolvedQName (rozstrzygnięta nazwa kolejki)	Brak	Pusty łańcuch lub odstępy
ResolvedQMgrNazwa (rozstrzygnięta nazwa menedżera kolejek)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_4.		
ObjectString (długa nazwa obiektu)	MQCHARV_DEFAULT	Zgodnie z definicją dla MQCHARV
SelectionString (łańcuch wyboru)	MQCHARV_DEFAULT	Zgodnie z definicją dla MQCHARV
ResObjectŁańcuch (przetłumaczona długa nazwa obiektu)	MQCHARV_DEFAULT	Zgodnie z definicją dla MQCHARV
ResolvedType (rozstrzygnięty typ obiektu)	MQOT_BRAK	0

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQOD_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQOD MyOD = {MQOD_DEFAULT};
```

Deklaracje językowe

Deklaracja w C dla MQOD

```
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 */
};
```

Deklaracja języka COBOL dla usługi MQOD

```
** MQOD structure
 10  MQOD.
** Structure identifier
 15  MQOD-STRUCID                    PIC X(4).
** Structure version number
 15  MQOD-VERSION                    PIC S9(9) BINARY.
** Object type
 15  MQOD-OBJECTTYPE                 PIC S9(9) BINARY.
** Object name
 15  MQOD-OBJECTNAME                 PIC X(48).
** Object queue manager name
 15  MQOD-OBJECTQMGRNAME             PIC X(48).
** Dynamic queue name
 15  MQOD-DYNAMICQNAME               PIC X(48).
```

```
** Alternate user identifier
 15  MQOD-ALTERNATEUSERID             PIC X(12).
** Number of object records present
 15  MQOD-RECSPRESENT                 PIC S9(9) BINARY.
** Number of local queues opened successfully
 15  MQOD-KNOWNDESTCOUNT              PIC S9(9) BINARY.
** Number of remote queues opened successfully
 15  MQOD-UNKNOWNDESTCOUNT            PIC S9(9) BINARY.
** Number of queues that failed to open
 15  MQOD-INVALIDDESTCOUNT            PIC S9(9) BINARY.
** Offset of first object record from start of MQOD
 15  MQOD-OBJECTRECOFFSET             PIC S9(9) BINARY.
** Offset of first response record from start of MQOD
 15  MQOD-RESPONSERECOFFSET           PIC S9(9) BINARY.
** Address of first object record
 15  MQOD-OBJECTRECPTR                POINTER.
** Address of first response record
 15  MQOD-RESPONSERECPTR              POINTER.
** Alternate security identifier
 15  MQOD-ALTERNATESECURITYID         PIC X(40).
** Resolved queue name
 15  MQOD-RESOLVEDQNAME               PIC X(48).
** Resolved queue manager name
 15  MQOD-RESOLVEDQMGRNAME            PIC X(48).
** Object Long name
 15 MQOD-OBJECTSTRING.
** Address of variable length string
    20  MQOD-OBJECTSTRING-VSPTR        POINTER.
** Offset of variable length string
    20  MQOD-OBJECTSTRING-VSOFFSET     PIC S9(9) BINARY.
** size of buffer
    20  MQOD-OBJECTSTRING-VSBUFSIZE    PIC S9(9) BINARY.
** Length of variable length string
    20  MQOD-OBJECTSTRING-VSLENGTH     PIC S9(9) BINARY.
** CCSID of variable length string
    20  MQOD-OBJECTSTRING-VSCCSID      PIC S9(9) BINARY.
** Message Selector
 15 MQOD-SELECTIONSTRING.
** Address of variable length string
    20  MQOD-SELECTIONSTRING-VSPTR     POINTER.
** Offset of variable length string
    20  MQOD-SELECTIONSTRING-VSOFFSET  PIC S9(9) BINARY.
** size of buffer
    20  MQOD-SELECTIONSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
    20  MQOD-SELECTIONSTRING-VSLENGTH  PIC S9(9) BINARY.
** CCSID of variable length string
    20  MQOD-SELECTIONSTRING-VSCCSID   PIC S9(9) BINARY.
** Resolved Long object name
 15 MQOD-RESOBJECTSTRING.
** Address of variable length string
    20  MQOD-RESOBJECTSTRING-VSPTR     POINTER.
** Offset of variable length string
    20  MQOD-RESOBJECTSTRING-VSOFFSET  PIC S9(9) BINARY.
** size of buffer
    20  MQOD-RESOBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
** Length of variable length string
    20  MQOD-RESOBJECTSTRING-VSLENGTH  PIC S9(9) BINARY.
** CCSID of variable length string
    20  MQOD-RESOBJECTSTRING-VSCCSID   PIC S9(9) BINARY.
** Alias queue resolved object type
    15  MQOD-RESOLVEDTYPE              PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla usługi MQOD

```
dcl
 1 MQOD based,
  3 StrucId             char(4),       /* Structure identifier */
  3 Version             fixed bin(31), /* Structure version number */
  3 ObjectType          fixed bin(31), /* Object type */
  3 ObjectName          char(48),      /* Object name */
  3 ObjectQMgrName      char(48),      /* Object queue manager name */
  3 DynamicQName        char(48),      /* Dynamic queue name */
  3 AlternateUserId     char(12),      /* Alternate user identifier */
  3 RecsPresent         fixed bin(31), /* Number of object records
                                          present */
  3 KnownDestCount      fixed bin(31), /* Number of local queues opened
                                          successfully */
  3 UnknownDestCount    fixed bin(31), /* Number of remote queues opened
```

```
                                       successfully */
  3 InvalidDestCount    fixed bin(31), /* Number of queues that failed to
                                       open */
  3 ObjectRecOffset     fixed bin(31), /* Offset of first object record
                                       from start of MQOD */
  3 ResponseRecOffset   fixed bin(31), /* Offset of first response record
                                       from start of MQOD */
  3 ObjectRecPtr        pointer,       /* Address of first object record */
  3 ResponseRecPtr      pointer,       /* Address of first response
                                       record */
  3 AlternateSecurityId char(40),      /* Alternate security identifier */
  3 ResolvedQName       char(48),      /* Resolved queue name */
  3 ResolvedQMgrName    char(48),      /* Resolved queue manager name */
  3 ObjectString,                      /* Object Long name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 SelectionString,                   /* Message Selection */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 ResObjectString,                   /* Resolved Long object name */
   5 VSPtr              pointer,       /* Address of variable length string */
   5 VSOffset           fixed bin(31), /* Offset of variable length string */
   5 VSBufSize          fixed bin(31), /* size of buffer */
   5 VSLength           fixed bin(31), /* Length of variable length string */
   5 VSCCSID            fixed bin(31), /* CCSID of variable length string */
  3 ResolvedType        fixed bin(31); /* Alias queue resolved object type */
```

Deklaracja High Level Assembler dla MQOD

```
MQOD                           DSECT
MQOD_STRUCID                   DS   CL4   Structure identifier
MQOD_VERSION                   DS   F     Structure version number
MQOD_OBJECTTYPE                DS   F     Object type
MQOD_OBJECTNAME                DS   CL48  Object name
MQOD_OBJECTQMGRNAME            DS   CL48  Object queue manager name
MQOD_DYNAMICQNAME              DS   CL48  Dynamic queue name
MQOD_ALTERNATEUSERID           DS   CL12  Alternate user identifier
MQOD_RECSPRESENT               DS   F     Number of object records present
MQOD_KNOWNDESTCOUNT            DS   F     Number of local queues opened
*                                         successfully
MQOD_UNKNOWNDESTCOUNT          DS   F     Number of remote queues opened
*                                         successfully
MQOD_INVALIDDESTCOUNT          DS   F     Number of queues that failed to
*                                         open
MQOD_OBJECTRECOFFSET           DS   F     Offset of first object record from
*                                         start of MQOD
MQOD_RESPONSERECOFFSET         DS   F     Offset of first response record
*                                         from start of MQOD
MQOD_OBJECTRECPTR              DS   F     Address of first object record
MQOD_RESPONSERECPTR            DS   F     Address of first response record
MQOD_ALTERNATESECURITYID       DS   XL40  Alternate security identifier
MQOD_RESOLVEDQNAME             DS   CL48  Resolved queue name
MQOD_RESOLVEDQMGRNAME          DS   CL48  Resolved queue manager name
MQOD_OBJECTSTRING              DS   F     Object Long name
MQOD_OBJECTSTRING_VSPTR        DS   F     Address of variable length string
MQOD_OBJECTSTRING_VSOFFSET     DS   F     Offset of variable length string
MQOD_OBJECTSTRING_VSBUFSIZE    DS   F     size of buffer
MQOD_OBJECTSTRING_VSLENGTH     DS   F     Length of variable length string
MQOD_OBJECTSTRING_VSCCSID      DS   F     CCSID of variable length string
MQOD_OBJECTSTRING_LENGTH       EQU  *- MQOD_OBJECTSTRING
                               ORG  MQOD_OBJECTSTRING
MQOD_OBJECTSTRING_AREA         DS   CL(MQOD_OBJECTSTRING_LENGTH)
*
MQOD_SELECTIONSTRING           DS   F     Message Selector
MQOD_SELECTIONSTRING_VSPTR     DS   F     Address of variable length string
MQOD_SELECTIONSTRING_VSOFFSET  DS   F     Offset of variable length string
MQOD_SELECTIONSTRING_VSBUFSIZE DS   F     size of buffer
MQOD_SELECTIONSTRING_VSLENGTH  DS   F     Length of variable length string
MQOD_SELECTIONSTRING_VSCCSID   DS   F     CCSID of variable length string
MQOD_SELECTIONSTRING_LENGTH    EQU  *- MQOD_SELECTIONSTRING
                               ORG  MQOD_SELECTIONSTRING
MQOD_SELECTIONSTRING_AREA      DS   CL(MQOD_SELECTIONSTRING_LENGTH)
*
```

```
MQOD_RESOBJECTSTRING           DS   F     Resolved Long object name
MQOD_RESOBJECTSTRING_VSPTR     DS   F     Address of variable length string
MQOD_RESOBJECTSTRING_VSOFFSET  DS   F     Offset of variable length string
MQOD_RESOBJECTSTRING_VSBUFSIZE DS   F     size of buffer
MQOD_RESOBJECTSTRING_VSLENGTH  DS   F     Length of variable length string
MQOD_RESOBJECTSTRING_VSCCSID   DS   F     CCSID of variable length string
MQOD_RESOBJECTSTRING_LENGTH    EQU  *- MQOD_RESOBJECTSTRING
                               ORG  MQOD_RESOBJECTSTRING
MQOD_RESOBJECTSTRING_AREA      DS   CL(MQOD_RESOBJECTSTRING_LENGTH)
MQOD_RESOLVEDTYPE              DS   F     Alias queue object resolved type
*
MQOD_LENGTH                    EQU  *-MQOD
                               ORG  MQOD
MQOD_AREA                      DS   CL(MQOD_LENGTH)
```

Deklaracja Visual Basic dla MQOD

```
Type MQOD
  StrucId             As String*4  'Structure identifier'
  Version             As Long      'Structure version number'
  ObjectType          As Long      'Object type'
  ObjectName          As String*48 'Object name'
  ObjectQMgrName      As String*48 'Object queue manager name'
  DynamicQName        As String*48 'Dynamic queue name'
  AlternateUserId     As String*12 'Alternate user identifier'
  RecsPresent         As Long      'Number of object records present'
  KnownDestCount      As Long      'Number of local queues opened'
                                   'successfully'
  UnknownDestCount    As Long      'Number of remote queues opened'
                                   'successfully'
  InvalidDestCount    As Long      'Number of queues that failed to'
                                   'open'
  ObjectRecOffset     As Long      'Offset of first object record from'
                                   'start of MQOD'
  ResponseRecOffset   As Long      'Offset of first response record'
                                   'from start of MQOD'
  ObjectRecPtr        As MQPTR     'Address of first object record'
  ResponseRecPtr      As MQPTR     'Address of first response record'
  AlternateSecurityId As MQBYTE40  'Alternate security identifier'
  ResolvedQName       As String*48 'Resolved queue name'
  ResolvedQMgrName    As String*48 'Resolved queue manager name'
End Type
```

StrucId (MQCHAR4) dla MQOD

Jest to identyfikator struktury struktury deskryptora obiektu. Jest to zawsze pole wejściowe. Jego wartością jest MQOD_STRUC_ID.

Wartość musi być następująca:

MQOD_STRUC_ID (identyfikator struktury MQD)
Identyfikator struktury deskryptora obiektu.

Dla języka programowania C zdefiniowana jest również stała MQOD_STRUC_ID_ARRAY. Ma taką samą wartość jak MQOD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQOD

Jest to numer wersji struktury; wartość musi być jedną z następujących wartości:

MQOD_VERSION_1
Struktura deskryptora obiektu Version-1 .

MQOD_VERSION_2
Struktura deskryptora obiektu Version-2 .

MQOD_VERSION_3
Struktura deskryptora obiektu Version-3 .

MQOD_VERSION_4
Struktura deskryptora obiektu Version-4 .

Wszystkie wersje są obsługiwane we wszystkich środowiskach IBM MQ V7.0 .

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQOD_CURRENT_VERSION
Bieżąca wersja struktury deskryptora obiektu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQOD_VERSION_1.

ObjectType (MQLONG) dla MQOD

Typ obiektu, którego nazwa znajduje się w deskryptorze obiektu. Dozwolone są następujące wartości:

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta. Nazwa obiektu znajduje się w polu *ObjectName* .

MQOT_Q
do kolejki błędów. Nazwa obiektu znajduje się w polu *ObjectName* .

LISTA NAZW MQOT_NAMELIST
Lista nazw. Nazwa obiektu znajduje się w polu *ObjectName*

PROCES_MQOT
Definicja procesu. Nazwa obiektu znajduje się w polu *ObjectName*

MQOT_Q_MGR
menedżerze kolejek. Nazwa obiektu znajduje się w polu *ObjectName*

TEMAT_MQOT
. Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ObjectName* i *ObjectString*.

Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQOT_Q.

ObjectName (MQCHAR48) dla usługi MQOD

Jest to nazwa lokalna obiektu zdefiniowana w menedżerze kolejek identyfikowanym przez *ObjectQMgrName*. Nazwa może zawierać następujące znaki:

- Wielkie litery (od A do Z)
- Małe litery (od a do z)
- Cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani wewnętrznych odstępów, ale może zawierać końcowe odstępy. Znak o kodzie zero oznacza koniec istotnych danych w nazwie; znak o kodzie zero i wszystkie następujące po nim znaki są traktowane jako odstępy. W podanych środowiskach obowiązują następujące ograniczenia:

- W systemach używających kodu EBCDIC Katakana nie można używać małych liter.
- W systemie z/OS:
  - Należy unikać nazw, które zaczynają się lub kończą znakiem podkreślenia; nie mogą być przetwarzane przez operacje i panele sterowania.
  - Znak procentu ma specjalne znaczenie dla RACF. Jeśli produkt RACF jest używany jako zewnętrzny menedżer zabezpieczeń, nazwy nie mogą zawierać procentu. Jeśli tak, nazwy te nie są uwzględniane podczas sprawdzania zabezpieczeń, gdy używane są profile ogólne RACF .
- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w cudzysłów, jeśli zostały podane w komendach. Te cudzysłowy nie mogą być podawane dla nazw, które występują jako pola w strukturach lub jako parametry w wywołaniach.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

Do wskazanych typów obiektów mają zastosowanie następujące punkty:

- Jeśli *ObjectName* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej i zwraca w polu *ObjectName* nazwę utworzonej kolejki. Kolejkę modelową można określić tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1 .
- Jeśli *ObjectName* jest nazwą kolejki aliasowej z TARGTYPE (TOPIC), najpierw wykonywane jest sprawdzenie zabezpieczeń w nazwanej kolejce aliasowej. Jest to normalne, gdy używane są kolejki aliasowe. Po pomyślnym zakończeniu sprawdzania zabezpieczeń wywołanie MQOPEN będzie kontynuowane i będzie zachowywać się jak wywołanie MQOPEN dla wywołania MQOT_TOPIC; obejmuje to sprawdzanie zabezpieczeń względem obiektu tematu administracyjnego.
- Jeśli *ObjectName* i *ObjectQMgrName* identyfikują kolejkę współużytkowaną, której właścicielem jest grupa współużytkowania kolejek, do której należy menedżer kolejek lokalnych, nie może istnieć również definicja kolejki o takiej samej nazwie w menedżerze kolejek lokalnych. Jeśli istnieje taka definicja (kolejka lokalna, kolejka aliasowa, kolejka zdalna lub kolejka modelowa), wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_NOT_UNIQUE.
- Jeśli otwierany obiekt jest listą dystrybucyjną (*RecsPresent* jest obecny i większy od zera), *ObjectName* musi być pusty lub musi być łańcuchem pustym. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_NAME_ERROR.
- Jeśli parametr *ObjectType* ma wartość MQOT_Q_MGR, mają zastosowanie reguły specjalne. W takim przypadku nazwa musi być całkowicie pusta aż do pierwszego znaku o kodzie zero lub końca pola.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ObjectName* jest nazwą kolejki modelowej, a we wszystkich innych przypadkach jest to pole tylko wejściowe. Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ObjectQMgrNazwa (MQCHAR48) dla MQOD

Jest to nazwa menedżera kolejek, w którym zdefiniowano obiekt *ObjectName* . Znaki poprawne w nazwie są takie same jak w przypadku nazwy *ObjectName* (patrz “ObjectName (MQCHAR48) dla usługi MQOD” na stronie 500). Nazwa, która jest całkowicie pusta aż do pierwszego znaku o kodzie zero lub końca pola, oznacza menedżera kolejek, z którym połączona jest aplikacja (menedżer kolejek lokalnych).

Do wskazanych typów obiektów mają zastosowanie następujące punkty:

- Jeśli *ObjectType* to MQOT_TOPIC, MQOT_NAMELIST, MQOT_PROCESS lub MQOT_Q_MGR, *ObjectQMgrName* musi być pusta lub musi być nazwą lokalnego menedżera kolejek.
- Jeśli *ObjectName* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej i zwraca w polu *ObjectQMgrName* nazwę menedżera kolejek, w którym została utworzona kolejka. Jest to nazwa lokalnego menedżera kolejek. Kolejkę modelową można określić tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1 .
- Jeśli parametr *ObjectName* jest nazwą kolejki klastra, a parametr *ObjectQMgrName* jest pusty, miejsce docelowe komunikatów wysłanych przy użyciu uchwytu kolejki zwróconego przez wywołanie MQOPEN jest wybierane przez menedżer kolejek (lub wyjście obciążenia klastra, jeśli zostało zainstalowane) w następujący sposób:
  - Jeśli określono opcję MQOO_BIND_ON_OPEN, menedżer kolejek wybiera konkretną instancję kolejki klastra podczas przetwarzania wywołania MQOPEN, a wszystkie komunikaty umieszczone za pomocą tego uchwytu kolejki są wysyłane do tej instancji.
  - Jeśli określono wartość MQOO_BIND_NOT_FIXED, menedżer kolejek może wybrać inną instancję kolejki docelowej (rezydującą w innym menedżerze kolejek w klastrze) dla każdego kolejnego wywołania MQPUT używającego tego uchwytu kolejki.

  Jeśli aplikacja musi wysłać komunikat do *konkretnej* instancji kolejki klastra (czyli instancji kolejki, która znajduje się w określonym menedżerze kolejek w klastrze), w polu *ObjectQMgrName* należy podać nazwę tego menedżera kolejek. Powoduje to, że lokalny menedżer kolejek wysyła komunikat do określonego docelowego menedżera kolejek.

- Jeśli *ObjectName* jest nazwą kolejki współużytkowanej, jest własnością grupy współużytkowania kolejek, do której należy lokalny menedżer kolejek, *ObjectQMgrName* może być nazwą grupy współużytkowania kolejek, nazwą lokalnego menedżera kolejek lub wartością pustą; komunikat jest umieszczany w tej samej kolejce, w zależności od tego, która z tych wartości jest określona.

  Grupy współużytkowania kolejek są obsługiwane tylko w systemie z/OS.
- Jeśli *ObjectName* jest nazwą kolejki współużytkowanej, której właścicielem jest zdalna grupa współużytkowania kolejek (grupa współużytkowania kolejek, do której nie należy lokalny menedżer kolejek), *ObjectQMgrName* musi być nazwą grupy współużytkowania kolejek. Można użyć nazwy menedżera kolejek, który należy do tej grupy, ale może to opóźnić komunikat, jeśli ten konkretny menedżer kolejek nie jest dostępny po nadejściu komunikatu do grupy współużytkowania kolejek.
- Jeśli otwierany obiekt jest listą dystrybucyjną (*RecsPresent* jest większe od zera), *ObjectQMgrName* musi być pusty lub musi być łańcuchem pustym. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_Q_MGR_NAME_ERROR.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ObjectName* jest nazwą kolejki modelowej, a we wszystkich innych przypadkach jest to pole tylko wejściowe. Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

DynamicQName (MQCHAR48) dla usługi MQOD

Jest to nazwa kolejki dynamicznej, która ma zostać utworzona przez wywołanie MQOPEN. Ma to znaczenie tylko wtedy, gdy parametr *ObjectName* określa nazwę kolejki modelowej; we wszystkich pozostałych przypadkach parametr *DynamicQName* jest ignorowany.

Znaki, które są poprawne w nazwie, są takie same jak w przypadku nazwy *ObjectName*, z tą różnicą, że gwiazdka jest również poprawna. Nazwa, która jest pusta (lub taka, w której tylko spacje występują przed pierwszym znakiem o kodzie zero), nie jest poprawna, jeśli *ObjectName* jest nazwą kolejki modelowej.

Jeśli ostatnim niepustym znakiem w nazwie jest gwiazdka (`*`), menedżer kolejek zastępuje gwiazdkę łańcuchem znaków, który gwarantuje, że nazwa wygenerowana dla kolejki jest unikalna w lokalnym menedżerze kolejek. Aby zapewnić wystarczającą liczbę znaków, gwiazdka jest poprawna tylko na pozycjach od 1 do 33. Po znaku gwiazdki nie mogą występować żadne znaki inne niż spacje ani znaki o kodzie zero.

Gwiazdka może występować na pierwszej pozycji znaku, w którym to przypadku nazwa składa się wyłącznie ze znaków wygenerowanych przez menedżer kolejek.

W systemie z/OSnie należy używać nazwy z gwiazdką na pierwszej pozycji znaku, ponieważ w kolejce z pełną nazwą, która jest generowana automatycznie, nie mogą być wykonywane żadne sprawdzenia zabezpieczeń.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Początkowa wartość tego pola jest określana przez środowisko:

- W systemie z/OSwartością jest `'CSQ.*'`.
- Na innych platformach wartością jest `'AMQ.*'`.

Wartością jest łańcuch zakończony znakiem o kodzie zero w języku C oraz łańcuch dopełniony znakiem o kodzie zero w innych językach programowania.

AlternateUserId (MQCHAR12) dla MQOD

Jeśli określono wartość MQOO_ALTERNATE_USER_AUTHORITY dla wywołania MQOPEN lub MQPMO_ALTERNATE_USER_AUTHORITY dla wywołania MQPUT1 , to pole zawiera alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla operacji otwierania zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona. Niektóre kontrole są jednak nadal przeprowadzane przy użyciu bieżącego identyfikatora użytkownika (na przykład sprawdzenia kontekstu).

Jeśli określono opcję MQOO_ALTERNATE_USER_AUTHORITY lub MQPMO_ALTERNATE_USER_AUTHORITY, a to pole jest całkowicie puste, aż do pierwszego znaku o kodzie zero lub końca pola, operacja otwierania może zakończyć się powodzeniem tylko wtedy, gdy do otwarcia tego obiektu z podanymi opcjami nie jest wymagana autoryzacja użytkownika.

Jeśli nie określono ani MQOO_ALTERNATE_USER_AUTHORITY, ani MQPMO_ALTERNATE_USER_AUTHORITY, to pole jest ignorowane.

W podanych środowiskach występują następujące różnice:

- W systemie z/OSdo sprawdzenia autoryzacji dla operacji otwierania używane jest tylko 8 pierwszych znaków pliku *AlternateUserId* . Jednak bieżący identyfikator użytkownika musi być autoryzowany do określenia tego konkretnego alternatywnego identyfikatora użytkownika; do tego sprawdzenia używane są wszystkie 12 znaków alternatywnego identyfikatora użytkownika. Identyfikator użytkownika może zawierać tylko znaki dozwolone przez zewnętrznego menedżera zabezpieczeń.

  Jeśli dla kolejki określono parametr *AlternateUserId* , wartość ta może być następnie używana przez menedżer kolejek podczas umieszczania komunikatów. Jeśli opcje MQPMO_ * _CONTEXT określone w wywołaniu MQPUT lub MQPUT1 powodują wygenerowanie przez menedżer kolejek informacji o kontekście tożsamości, menedżer kolejek umieszcza wartość *AlternateUserId* w polu *UserIdentifier* w deskryptorze MQMD komunikatu zamiast bieżącego identyfikatora użytkownika.
- W innych środowiskach parametr *AlternateUserId* jest używany tylko do sprawdzania praw dostępu dla otwieranego obiektu. Jeśli obiekt jest kolejką, parametr *AlternateUserId* nie ma wpływu na zawartość pola *UserIdentifier* w deskryptorze MQMD komunikatów wysłanych przy użyciu tego uchwytu kolejki.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 12 znaków odstępu w innych językach programowania.

RecsPresent (MQLONG) dla MQOD

Jest to liczba rekordów obiektów MQOR, które zostały udostępnione przez aplikację. Jeśli ta liczba jest większa od zera, oznacza to, że lista dystrybucyjna jest otwierana, a *RecsPresent* jest liczbą kolejek docelowych na liście. Lista dystrybucyjna może zawierać tylko jedno miejsce docelowe.

Wartość *RecsPresent* nie może być mniejsza niż zero, a jeśli jest większa niż zero *ObjectType* musi być równa MQOT_Q; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RECS_PRESENT_ERROR, jeśli te warunki nie są spełnione.

W systemie z/OSto pole musi mieć wartość zero.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_2.

KnownDest-liczba (MQLONG) dla MQOD

Jest to liczba kolejek na liście dystrybucyjnej, które są tłumaczone na kolejki lokalne i które zostały pomyślnie otwarte. Liczba ta nie obejmuje kolejek, które są tłumaczone na kolejki zdalne (nawet jeśli początkowo do przechowywania komunikatu używana jest lokalna kolejka transmisji). Jeśli istnieje, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_1.

UnknownDestLiczba (MQLONG) dla MQOD

Jest to liczba kolejek na liście dystrybucyjnej, które są tłumaczone na kolejki zdalne i które zostały pomyślnie otwarte. Jeśli istnieje, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_1.

Liczba operacji InvalidDest(MQLONG) dla MQOD

Jest to liczba kolejek na liście dystrybucyjnej, których otwarcie nie powiodło się. Jeśli istnieje, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Uwaga: Jeśli to pole jest obecne, jest ono ustawiane tylko wtedy, gdy parametr **CompCode** w wywołaniu MQOPEN lub MQPUT1 ma wartość MQCC_OK lub MQCC_WARNING; nie jest ustawiane, jeśli parametr **CompCode** ma wartość MQCC_FAILED.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_1.

ObjectRecPrzesunięcie (MQLONG) dla MQOD

Jest to przesunięcie w bajtach pierwszego rekordu obiektu MQOR od początku struktury MQOD. Przesunięcie może być dodatnie lub ujemne. Opcja *ObjectRecOffset* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Podczas otwierania listy dystrybucyjnej należy podać tablicę zawierającą jeden lub więcej rekordów obiektu MQOR, aby określić nazwy kolejek docelowych na liście dystrybucyjnej. Można to zrobić na jeden z dwóch sposobów:

- Używając zmiennej przesunięcia *ObjectRecOffset*.

  W takim przypadku aplikacja musi zadeklarować własną strukturę zawierającą element MQOD, po którym następuje tablica rekordów MQOR (z wymaganą liczbą elementów tablicy), i ustawić parametr *ObjectRecOffset* na przesunięcie pierwszego elementu tablicy od początku elementu MQOD. Upewnij się, że to przesunięcie jest poprawne i ma wartość, która może być ustawiona w MQLONG (najbardziej restrykcyjnym językiem programowania jest COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

  Języka *ObjectRecOffset* należy używać w przypadku języków programowania, które nie obsługują typu danych wskaźnika lub implementują typ danych wskaźnika w sposób, który nie jest przenośny w różnych środowiskach (na przykład w języku programowania COBOL).

- Używając pola wskaźnika *ObjectRecPtr*.

  W takim przypadku aplikacja może zadeklarować tablicę struktur MQOR niezależnie od struktury MQOD i ustawić parametr *ObjectRecPtr* na adres tablicy.

  ObjectRecPtr służy do obsługi języków programowania obsługujących typ danych wskaźnika w sposób przenośny w różnych środowiskach (na przykład w języku programowania C).

Niezależnie od wybranej techniki, należy użyć jednej z następujących metod: *ObjectRecOffset* i *ObjectRecPtr* ; Wywołanie nie powiodło się z kodem przyczyny MQRC_OBJECT_RECORDS_ERROR, jeśli obie wartości są równe zero lub obie są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_2.

ResponseRecprzesunięcie (MQLONG) dla MQOD

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQOD. Przesunięcie może być dodatnie lub ujemne. Opcja *ResponseRecOffset* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Podczas otwierania listy dystrybucyjnej można udostępnić tablicę zawierającą jeden lub więcej rekordów odpowiedzi MQRR w celu zidentyfikowania kolejek, których otwarcie nie powiodło się (pole*CompCode* w MQRR), oraz przyczyny każdego niepowodzenia (pole*Reason* w MQRR). Dane są zwracane w tablicy rekordów odpowiedzi w tej samej kolejności, w jakiej nazwy kolejek występują w tablicy rekordów obiektów. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (tzn. niektóre kolejki zostały pomyślnie otwarte, podczas gdy inne nie powiodły się, lub wszystkie nie powiodły się, ale z różnych przyczyn). Kod przyczyny MQRC_MULTIPLE_REASON z wywołania wskazuje ten przypadek. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna jest

zwracana w parametrze **Reason** wywołania MQOPEN lub MQPUT1 , a rekordy odpowiedzi nie są ustawione. Rekordy odpowiedzi są opcjonalne, ale jeśli zostaną podane, musi być ich *RecsPresent* .

Rekordy odpowiedzi mogą być udostępniane w taki sam sposób, jak rekordy obiektów, przez określenie przesunięcia w *ResponseRecOffset*lub przez określenie adresu w *ResponseRecPtr* ; Szczegółowe informacje na ten temat zawiera sekcja "ObjectRecPrzesunięcie (MQLONG) dla MQOD" na stronie 504. Można jednak użyć tylko jednego z następujących parametrów: *ResponseRecOffset* i *ResponseRecPtr* . Wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_RESPONSE_RECORDS_ERROR, jeśli oba te parametry są niezerowe.

W przypadku wywołania MQPUT1 te rekordy odpowiedzi są używane do zwracania informacji o błędach, które wystąpiły podczas wysyłania komunikatu do kolejek na liście dystrybucyjnej, a także o błędach, które wystąpiły podczas otwierania kolejek. Kod zakończenia i kod przyczyny z operacji umieszczania dla kolejki zastępują te z operacji otwierania dla tej kolejki tylko wtedy, gdy kod zakończenia z tej ostatniej to MQCC_OK lub MQCC_WARNING.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_2.

ObjectRecPtr (MQPTR) dla MQOD

Jest to adres pierwszego rekordu obiektu MQOR. Opcja *ObjectRecPtr* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Do określenia rekordów obiektów można użyć wartości *ObjectRecPtr* lub *ObjectRecOffset* , ale nie obu tych wartości; opis pola *ObjectRecOffset* znajduje się w sekcji "ObjectRecPrzesunięcie (MQLONG) dla MQOD" na stronie 504. Jeśli nie jest używany parametr *ObjectRecPtr*, należy ustawić go na wskaźnik pusty lub bajty puste.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_2.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, to pole jest deklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null.

ResponseRecPtr (MQPTR) dla MQOD

Jest to adres pierwszego rekordu odpowiedzi MQRR. Opcja *ResponseRecPtr* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Do określenia rekordów odpowiedzi należy użyć wartości *ResponseRecPtr* lub *ResponseRecOffset* , ale nie obu tych wartości. Szczegółowe informacje na ten temat zawiera sekcja "ResponseRecprzesunięcie (MQLONG) dla MQOD" na stronie 504. Jeśli nie jest używany parametr *ResponseRecPtr*, należy ustawić go na wskaźnik pusty lub bajty puste.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_2.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, to pole jest deklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null.

AlternateSecurityId (MQBYTE40) dla MQOD

Jest to identyfikator bezpieczeństwa, który jest przekazywany z *AlternateUserId* do usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji. Parametr *AlternateSecurityId* jest używany tylko wtedy, gdy:

- W wywołaniu MQOPEN określono uprawnienie MQOO_ALTERNATE_USER_AUTHORITY lub
- Parametr MQPMO_ALTERNATE_USER_AUTHORITY jest określony w wywołaniu MQPUT1 .

i pole *AlternateUserId* nie jest całkowicie puste aż do pierwszego znaku o kodzie zero lub końca pola.

W systemie Windowsmożna użyć identyfikatora *AlternateSecurityId* do podania identyfikatora zabezpieczeń Windows (SID), który jednoznacznie identyfikuje *AlternateUserId*. Identyfikator SID użytkownika można uzyskać z systemu Windows za pomocą wywołania interfejsu API produktu LookupAccountName( ) Windows .

W systemie z/OSto pole jest ignorowane.

Pole *AlternateSecurityId* ma następującą strukturę:

- Pierwszy bajt jest binarną liczbą całkowitą zawierającą długość istotnych danych, które następują po nim; wartość ta wyklucza sam bajt długości. Jeśli nie ma identyfikatora zabezpieczeń, długość wynosi zero.
- Drugi bajt wskazuje typ identyfikatora zabezpieczeń, który jest obecny; możliwe są następujące wartości:

  MQSIDT_NT_SECURITY_ID
  Identyfikator zabezpieczeń Windows .

  MQSIDT_BRAK
  Brak identyfikatora zabezpieczeń.
- Trzeci i kolejny bajt do długości określonej przez pierwszy bajt zawiera sam identyfikator bezpieczeństwa.
- Pozostałe bajty w polu są ustawione na zero binarne.

Można użyć następującej wartości specjalnej:

MQSID_BRAK
Nie określono identyfikatora zabezpieczeń.

Wartością długości pola jest zero binarne.

W przypadku języka programowania C zdefiniowana jest również stała MQSID_NONE_ARRAY. Ma ona taką samą wartość jak parametr MQSID_NONE, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_SECURITY_ID_LENGTH. Wartością początkową tego pola jest MQSID_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_3.

ResolvedQName (MQCHAR48) dla usługi MQOD

Jest to nazwa kolejki docelowej po przetłumaczeniu nazwy przez lokalny menedżer kolejek. Zwracana nazwa jest nazwą kolejki, która istnieje w menedżerze kolejek identyfikowanym przez *ResolvedQMgrName*.

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą do przeglądania, wprowadzania lub wyprowadzania (lub dowolnej kombinacji). Jeśli otwierany jest dowolny z następujących obiektów, parametr *ResolvedQName* jest ustawiany na wartość pustą:

- To nie jest kolejka
- Kolejka, ale nie otwarta do przeglądania, wejścia lub wyjścia
- Lista dystrybucyjna
- Kolejka aliasowa, która odwołuje się do obiektu tematu (zamiast tego należy odwołać się do łańcuchaResObject).
- Kolejka aliasowa, która jest tłumaczona na obiekt tematu.

Jest to pole wyjściowe. Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_3.

ResolvedQMgrNazwa (MQCHAR48) dla MQOD

Jest to nazwa docelowego menedżera kolejek po przetłumaczeniu nazwy przez lokalny menedżer kolejek. Zwracana nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez *ResolvedQName*. *ResolvedQMgrName* może być nazwą lokalnego menedżera kolejek.

Jeśli *ResolvedQName* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejek, do której należy lokalny menedżer kolejek, *ResolvedQMgrName* jest nazwą grupy współużytkowania kolejek. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, parametr *ResolvedQName* może być nazwą grupy współużytkowania kolejek lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejek (rodzaj zwracanej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą do przeglądania, wprowadzania lub wyprowadzania (lub dowolnej kombinacji). Jeśli otwierany jest dowolny z następujących obiektów, parametr *ResolvedQMgrName* jest ustawiany na wartość pustą:

- To nie jest kolejka
- Kolejka, ale nie otwarta do przeglądania, wejścia lub wyjścia
- Kolejka klastra z podaną wartością MQOO_BIND_NOT_FIXED (lub z wartością MQOO_BIND_AS_Q_DEF, gdy atrybut kolejki **DefBind** ma wartość MQBND_BIND_NOT_FIXED)
- Lista dystrybucyjna

Jest to pole wyjściowe. Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQOD_VERSION_3.

ObjectString (MQCHARV) dla MQOD

Pole ObjectString określa długą nazwę obiektu.

Określa długą nazwę obiektu, która ma być używana. To pole jest przywoływane tylko w przypadku niektórych wartości zmiennej *ObjectType*i jest ignorowane w przypadku wszystkich innych wartości. Szczegółowe informacje o tym, które wartości wskazują, że to pole jest używane, można znaleźć w opisie pola *ObjectType* .

Jeśli parametr *ObjectString* zostanie podany niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekroczy ona maksymalną długość, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_OBJECT_STRING_ERROR.

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

SelectionString (MQCHARV) dla usługi MQOD

Jest to łańcuch używany do określania kryteriów wyboru używanych podczas pobierania komunikatów z kolejki.

Parametr *SelectionString* nie może być podany w następujących przypadkach:

- Jeśli parametr *ObjectType* nie ma wartości MQOT_Q
- Jeśli otwierana kolejka nie jest otwierana przy użyciu jednej z opcji MQOO_BROWSE lub MQOO_INPUT_*

Jeśli w takich przypadkach zostanie podana wartość *SelectionString* , wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_SELECTOR_INVALID_FOR_TYPE.

Jeśli parametr *SelectionString* jest określony niepoprawnie, zgodnie z opisem sposobu użycia struktury “MQCHARV-łańcuch o zmiennej długości” na stronie 299 lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SELECTION_STRING_ERROR. Maksymalna długość właściwości *SelectionString* wynosi MQ_SELECTOR_LENGTH.

Użycie *SelectionString* jest opisane w sekcji Selektory.

ResObjectŁańcuch (MQCHARV) dla MQOD

Pole łańcucha ResObjectjest długą nazwą obiektu po tym, jak menedżer kolejek rozstrzyga nazwę podaną w polu *ObjectName* .

To pole jest zwracane tylko dla tematów i aliasów kolejek, które odwołują się do obiektu tematu.

Jeśli w polu *ObjectString* zostanie podana długa nazwa obiektu i w polu *ObjectName*nie zostanie podana żadna wartość, wartość zwracana w tym polu będzie taka sama, jak w polu *ObjectString*.

Jeśli to pole zostanie pominięte (czyli pole ResObjectString.VSBufSize ma wartość zero), pole *ResObjectString* nie zostanie zwrócone, ale długość zostanie zwrócona w polu ResObjectString.VSLength.

Jeśli długość buforu (podana w pliku ResObjectStrng.VSBufSize) jest mniejsza niż pełna wielkość *ResObjectString*, łańcuch zostanie obcięty i zwróci tyle znaków po prawej stronie, ile może zmieścić się w podanym buforze.

Jeśli parametr *ResObjectString* zostanie podany niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekroczy ona maksymalną długość, wywołanie nie powiedzie się z kodem przyczyny MQRC_RES_OBJECT_STRING_ERROR.

ResolvedType (MQLONG) dla usługi MQOD

Typ otwieranego obiektu rozstrzygniętego (podstawowego).

Możliwe wartości:

MQOT_Q
Rozstrzygnięty obiekt jest kolejką. Ta wartość ma zastosowanie, gdy kolejka jest otwierana bezpośrednio lub gdy kolejka aliasowa wskazuje kolejkę.

TEMAT_MQOT
Rozstrzygnięty obiekt jest tematem. Ta wartość ma zastosowanie, gdy temat jest otwierany bezpośrednio lub gdy otwierana jest kolejka aliasowa wskazująca obiekt tematu.

MQOT_BRAK
Rozstrzygnięty typ nie jest ani kolejką, ani tematem.

MQOR-rekord obiektu

Struktura MQOR służy do określania nazwy kolejki i nazwy menedżera kolejek dla pojedynczej kolejki docelowej. MQOR jest strukturą wejściową dla wywołań MQOPEN i MQPUT1 .

Dostępność

Struktura MQOR jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

Zestaw znaków i kodowanie

Dane w MQOR muszą być w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Użycie

Udostępniając tablicę tych struktur w wywołaniu MQOPEN, można otworzyć listę kolejek; ta lista jest nazywana listą dystrybucyjną. Każdy komunikat umieszczony za pomocą uchwytu kolejki zwróconego przez wywołanie MQOPEN jest umieszczany w każdej kolejce na liście, pod warunkiem, że kolejka została pomyślnie otwarta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 505. Pola w MQOR dla MQOR

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
ObjectName (nazwa obiektu)	Brak	Pusty łańcuch lub odstępy
ObjectQMgrNazwa (nazwa menedżera kolejek obiektu)	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
2. W języku programowania C: zmienna makraMQOR_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQOR MyOR = {MQOR_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQOR

```
typedef struct tagMQOR MQOR;
struct tagMQOR {
  MQCHAR48  ObjectName;      /* Object name */
  MQCHAR48  ObjectQMgrName;  /* Object queue manager name */
};
```

Deklaracja COBOL dla MQOR

```
**   MQOR structure
  10 MQOR.
**    Object name
   15 MQOR-OBJECTNAME     PIC X(48).
**    Object queue manager name
   15 MQOR-OBJECTQMGRNAME PIC X(48).
```

Deklaracja PL/I dla MQOR

```
dcl
 1 MQOR based,
  3 ObjectName     char(48), /* Object name */
  3 ObjectQMgrName char(48); /* Object queue manager name */
```

Deklaracja Visual Basic dla MQOR

```
Type MQOR
  ObjectName     As String*48 'Object name'
  ObjectQMgrName As String*48 'Object queue manager name'
End Type
```

ObjectName (MQCHAR48) dla MQOR

Jest to to samo pole, co pole *ObjectName* w strukturze MQOD (szczegółowe informacje zawiera dokument MQOD), z tą różnicą, że:

- Musi to być nazwa kolejki.
- Nie może to być nazwa kolejki modelowej.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ObjectQMgrNazwa (MQCHAR48) dla MQOR

Jest to takie samo jak pole *ObjectQMgrName* w strukturze MQOD (szczegółowe informacje zawiera sekcja MQOD).

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

MQPD-deskryptor właściwości

Struktura **MQPD** jest używana do definiowania atrybutów właściwości. Struktura jest parametrem wejścia/wyjścia w wywołaniu MQSETMP i parametrem wyjściowym w wywołaniu MQINQMP.

Dostępność

Struktura **MQPD** jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

i dla IBM MQ MQI clients.

Zestaw znaków i kodowanie

Dane w pliku **MQPD** muszą znajdować się w zestawie znaków aplikacji i kodowania aplikacji (**MQENC_NATIVE**).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 506. Pola w MQPD

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQPD_STRUC_ID	'PD'
Wersja (numer wersji struktury)	MQPD_VERSION_1	1
Opcje (opcje)	MQPD_BRAK	0
Wsparcie (wymagana obsługa właściwości komunikatu)	MQPD_SUPPORT_OPTIONAL	0
Kontekst (kontekst komunikatu, do którego należy właściwość)	MQPD_NO_CONTEXT	0
CopyOptions (opcje kopiowania, do których należy właściwość)	MQCOPY_DEFAULT	0

Uwagi:

1. W języku programowania C zmienna makra MQPD_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQPD MyPD = {MQPD_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQPD

```
typedef struct tagMQPD MQPD;
struct tagMQPD {
  MQCHAR4  StrucId;     /* Structure identifier */
  MQLONG   Version;     /* Structure version number */
  MQLONG   Options;     /* Options that control the action of
                           MQSETMP and MQINQMP */
  MQLONG   Support;     /* Property support option */
  MQLONG   Context;     /* Property context */
  MQLONG   CopyOptions; /* Property copy options */
};
```

Deklaracja języka COBOL dla MQPD

```
**   MQPD structure
  10 MQPD.
**    Structure identifier
   15 MQPD-STRUCID PIC X(4).
**    Structure version number
   15 MQPD-VERSION PIC S9(9) BINARY.
**    Options that control the action of MQSETMP and
**    MQINQMP
   15 MQPD-OPTIONS PIC S9(9) BINARY.
**    Property support option
   15 MQPD-SUPPORT PIC S9(9) BINARY.
**    Property context
   15 MQPD-CONTEXT PIC S9(9) BINARY.
**    Property copy options
   15 MQPD-COPYOPTIONS PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQPD

```
dcl
 1 MQPD based,
  3 StrucId     char(4),       /* Structure identifier */
  3 Version     fixed bin(31), /* Structure version number */
```

```
  3 Options     fixed bin(31), /* Options that control the action
                                  of MQSETMP and MQINQMP */
  3 Support     fixed bin(31), /* Property support option */
  3 Context     fixed bin(31), /* Property context */
  3 CopyOptions fixed bin(31); /* Property copy options */
```

Deklaracja High Level Assembler dla MQPD

```
MQPD             DSECT
MQPD_STRUCID     DS   CL4    Structure identifier
MQPD_VERSION     DS   F      Structure version number
MQPD_OPTIONS     DS   F      Options that control the
*                            action of MQSETMP and MQINQMP
MQPD_SUPPORT     DS   F      Property support option
MQPD_CONTEXT     DS   F      Property context
MQPD_COPYOPTIONS DS   F      Property copy options
MQPD_LENGTH      EQU  *-MQPD
MQPD_AREA        DS   CL(MQPD_LENGTH)
```

StrucId (MQCHAR4) dla MQPD

Jest to identyfikator struktury struktury deskryptora właściwości. Jest to zawsze pole wejściowe. Jego wartością jest MQPD_STRUC_ID.

Wartość musi być następująca:

MQPD_STRUC_ID
Identyfikator struktury deskryptora właściwości.

Dla języka programowania C zdefiniowana jest również stała **MQPD_STRUC_ID_ARRAY** . Ma taką samą wartość jak **MQPD_STRUC_ID**, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQPD

Jest to numer wersji struktury; wartość musi być następująca:

MQPD_VERSION_1
Struktura deskryptora właściwości Version-1 .

Następująca stała określa numer wersji bieżącej:

MQPD_CURRENT_VERSION (MQPD_CURRENT_VERSION)
Bieżąca wersja struktury deskryptora właściwości.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **MQPD_VERSION_1**.

Opcje (MQLONG) dla MQPD

Wartość musi być następująca:

MQPD_BRAK
Nie określono żadnych opcji

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQPD_NONE.

Obsługa (MQLONG) dla MQPD

W tym polu opisano, jaki poziom obsługi właściwości komunikatu jest wymagany przez menedżer kolejek, aby komunikat zawierający tę właściwość został umieszczony w kolejce. Dotyczy to tylko właściwości zdefiniowanych przez IBM MQ; obsługa wszystkich innych właściwości jest opcjonalna.

Pole jest automatycznie ustawiane na poprawną wartość, gdy właściwość zdefiniowana przez program IBM MQjest znana menedżerowi kolejek. Jeśli właściwość nie zostanie rozpoznana, zostanie przypisana wartość MQPD_SUPPORT_OPTIONAL. Gdy menedżer kolejek odbiera komunikat zawierający właściwość zdefiniowaną przez IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną, poprawia wartość pola *Support* .

Podczas ustawiania właściwości zdefiniowanej przez IBM MQza pomocą wywołania MQSETMP dla uchwytu komunikatu, w którym ustawiono opcję MQCMHO_NO_VALIDATION, *Support* staje się polem wejściowym. Dzięki temu aplikacja może umieścić właściwość zdefiniowaną w programie IBM MQo poprawnej wartości, jeśli właściwość nie jest obsługiwana przez połączony menedżer kolejek, ale komunikat ma być przetwarzany w innym menedżerze kolejek.

Wartość MQPD_SUPPORT_OPTIONAL jest zawsze przypisywana do właściwości, które nie są właściwościami zdefiniowanymi przez IBM MQ.

Jeśli menedżer kolejek produktu IBM WebSphere MQ 7.0 , który obsługuje właściwości komunikatu, odbiera właściwość zawierającą nierozpoznaną wartość *Support* , właściwość ta jest traktowana tak, jakby:

- Parametr MQPD_SUPPORT_REQUIRED został określony, jeśli którakolwiek z nierozpoznanych wartości jest zawarta w parametrze MQPD_REJECT_UNSUP_MASK.
- Określono MQPD_SUPPORT_REQUIRED_IF_LOCAL, jeśli którakolwiek z nierozpoznanych wartości jest zawarta w masce MQPD_ACCEPT_UNSUP_IF_XMIT_MASK
- W przeciwnym razie podano parametr MQPD_SUPPORT_OPTIONAL.

Jedna z następujących wartości jest zwracana przez wywołanie MQINQMP lub jedna z wartości może zostać określona, gdy jest używane wywołanie MQSETMP dla uchwytu komunikatu, w którym ustawiono opcję MQCMHO_NO_VALIDATION:

MQPD_SUPPORT_OPTIONAL
Właściwość jest akceptowana przez menedżer kolejek, nawet jeśli nie jest obsługiwana. Właściwość można odrzucić, aby komunikat przepływał do menedżera kolejek, który nie obsługuje właściwości komunikatu. Ta wartość jest również przypisywana do właściwości, które nie są zdefiniowane w systemie IBM MQ.

MQPD_SUPPORT_REQUIRED,
Obsługa tej właściwości jest wymagana. Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje właściwości zdefiniowanej przez IBM MQ. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_UNSUPPORTED_PROPERTY.

MQPD_SUPPORT_REQUIRED_IF_LOCAL
Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje właściwości zdefiniowanej przez IBM MQ, jeśli komunikat jest przeznaczony dla kolejki lokalnej. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_UNSUPPORTED_PROPERTY.

Wywołanie MQPUT lub MQPUT1 powiedzie się, jeśli komunikat jest przeznaczony dla zdalnego menedżera kolejek.

Jest to pole wyjściowe wywołania MQINQMP i pole wejściowe wywołania MQSETMP, jeśli uchwyt komunikatu został utworzony z ustawioną opcją MQCMHO_NO_VALIDATION. Wartością początkową tego pola jest MQPD_SUPPORT_OPTIONAL.

Kontekst (MQLONG) dla MQPD

Opisuje kontekst komunikatu, do którego należy właściwość.

Gdy menedżer kolejek odbiera komunikat zawierający właściwość zdefiniowaną przez IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną, poprawia wartość pola *Context* .

Można podać następującą opcję:

MQPD_USER_CONTEXT
Właściwość jest powiązana z kontekstem użytkownika.

Aby można było ustawić właściwość powiązaną z kontekstem użytkownika za pomocą wywołania MQSETMP, nie jest wymagana żadna specjalna autoryzacja.

W menedżerze kolejek systemu IBM WebSphere MQ 7.0 właściwość powiązana z kontekstem użytkownika jest zapisywana w sposób opisany w sekcji MQOO_SAVE_ALL_CONTEXT. Wywołanie

MQPUT z określonym parametrem MQPMO_PASS_ALL_CONTEXT powoduje skopiowanie właściwości z zapisanego kontekstu do nowego komunikatu.

Jeśli opisana wcześniej opcja nie jest wymagana, można użyć następującej opcji:

MQPD_NO_CONTEXT
Właściwość nie jest powiązana z kontekstem komunikatu.

Nierozpoznana wartość została odrzucona z kodem *Reason* MQRC_PD_ERROR

Jest to pole wejściowe/wyjściowe wywołania MQSETMP i pole wyjściowe wywołania MQINQMP. Wartością początkową tego pola jest MQPD_NO_CONTEXT.

CopyOptions (MQLONG) dla MQPD

Opisuje typ komunikatów, do których właściwość powinna zostać skopiowana. Jest to pole wyjściowe tylko dla rozpoznawanych IBM MQ zdefiniowanych właściwości; IBM MQ ustawia odpowiednią wartość.

Gdy menedżer kolejek odbiera komunikat zawierający właściwość zdefiniowaną przez IBM MQ , którą menedżer kolejek rozpoznaje jako niepoprawną, poprawia wartość pola *CopyOptions* .

Można określić jedną lub więcej z tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

MQCOPY_FORWARD
Ta właściwość jest kopiowana do przekazywanej wiadomości.

MQCOPY_PUBLISH
Ta właściwość jest kopiowana do komunikatu odebranego przez subskrybenta podczas publikowania komunikatu.

MQCOPY_REPLY
Ta właściwość jest kopiowana do komunikatu odpowiedzi.

MQCOPY_REPORT
Ta właściwość jest kopiowana do komunikatu raportu.

MQCOPY_ALL
Ta właściwość jest kopiowana do wszystkich typów kolejnych komunikatów.

Opcja domyślna: Można podać następującą opcję, aby określić domyślny zestaw opcji kopiowania:

MQCOPY_DEFAULT
Ta właściwość jest kopiowana do przekazywanego komunikatu, do komunikatu raportu lub do komunikatu odebranego przez subskrybent podczas publikowania komunikatu.

Jest to równoważne z określeniem kombinacji opcji MQCOPY_FORWARD, MQCOPY_REPORT i MQCOPY_PUBLISH.

Jeśli żadna z opisanych wcześniej opcji nie jest wymagana, należy użyć następującej opcji:

MQCOPY_NONE
Ta wartość wskazuje, że nie określono żadnych innych opcji kopiowania; programowo nie istnieje relacja między tą właściwością a kolejnymi komunikatami. Ta wartość jest zawsze zwracana dla właściwości deskryptora komunikatu.

Jest to pole wejściowe/wyjściowe wywołania MQSETMP i pole wyjściowe wywołania MQINQMP. Wartością początkową tego pola jest MQCOPY_DEFAULT.

MQPMO-opcje umieszczania komunikatów

Struktura MQPMO umożliwia aplikacji określanie opcji sterujących umieszczaniem komunikatów w kolejkach lub publikowaniem ich w tematach. Struktura jest parametrem wejścia/wyjścia w wywołaniach MQPUT i MQPUT1 .

Wersja

Bieżąca wersja programu MQPMO to MQPMO_VERSION_3. Niektóre pola są dostępne tylko w niektórych wersjach programu MQPMO. Jeśli konieczne jest przeniesienie aplikacji między kilkoma środowiskami, należy upewnić się, że wersja produktu MQPMO jest spójna we wszystkich środowiskach. Pola, które istnieją tylko w określonych wersjach struktury, są identyfikowane jako takie w tym temacie i w opisach pól.

Pliki nagłówkowe, COPY i INCLUDE udostępnione dla obsługiwanych języków programowania zawierają najnowszą wersję produktu MQPMO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *Version* ustawioną na MQPMO_VERSION_1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *Version* numer wersji wymaganej wersji.

Zestaw znaków i kodowanie

Dane w MQPMO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 507. Pola w MQPMO

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQPMO_STRUC_ID,	'PMO¬'
Wersja (numer wersji struktury)	MQPMO_VERSION_1	1
Opcje (opcje sterujące działaniem komend MQPUT i MQPUT1)	MQPMO_BRAK	0
Limit czasu (zarezerwowany)	Brak	-1
Kontekst (uchwyt obiektu kolejki wejściowej)	Brak	0
KnownDestCount (liczba komunikatów pomyślnie wysłanych do kolejek lokalnych)	Brak	0
UnknownDestLiczba (liczba komunikatów pomyślnie wysłanych do kolejek zdalnych)	Brak	0
InvalidDestLiczba (liczba komunikatów, których nie można było wysłać)	Brak	0
ResolvedQName (rozstrzygnięta nazwa kolejki docelowej)	Brak	Pusty łańcuch lub odstępy
ResolvedQMgrNazwa (rozstrzygnięta nazwa docelowego menedżera kolejek)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.		
RecsPresent (liczba rekordów komunikatów umieszczania lub rekordów odpowiedzi)	Brak	0
PutMsgRecFields (flagi wskazujące, które pola MQPMR są obecne)	MQPMRF_BRAK	0

Tabela 507. Pola w MQPMO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
PutMsgRecOffset (przesunięcie pierwszego rekordu komunikatu umieszczania od początku MQPMO)	Brak	0
ResponseRecprzesunięcie (przesunięcie pierwszego rekordu odpowiedzi od początku MQPMO)	Brak	0
PutMsgRecPtr (adres pierwszego rekordu komunikatu umieszczania)	Brak	Pusty wskaźnik lub puste bajty
ResponseRecPtr (adres pierwszego rekordu odpowiedzi)	Brak	Pusty wskaźnik lub puste bajty
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_3.		
OriginalMsgUchwyt (oryginalny uchwyt komunikatu)	MQHM_NONE	0
NewMsg (nowy uchwyt komunikatu)	MQHM_NONE	0
Działanie (typ wykonywanej operacji umieszczania i relacja między oryginalnym komunikatem określonym w polu *OriginalMsgHandle* a nowym komunikatem określonym w polu *NewMsgHandle*)	MQACTP_NOWA	0
PubLevel (poziom subskrypcji, której dotyczy publikacja)	Brak	9

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQPMO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQPMO MyPMO = {MQPMO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQPMO

```
typedef struct tagMQPMO MQPMO;
struct tagMQPMO {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQLONG    Options;            /* Options that control the action of
                                   MQPUT and MQPUT1 */
  MQLONG    Timeout;            /* Reserved */
  MQHOBJ    Context;            /* Object handle of input queue */
  MQLONG    KnownDestCount;     /* Number of messages sent
                                   successfully to local queues */
  MQLONG    UnknownDestCount;   /* Number of messages sent
                                   successfully to remote queues */
  MQLONG    InvalidDestCount;   /* Number of messages that could not
```

```
                                   be sent */
  MQCHAR48  ResolvedQName;      /* Resolved name of destination
                                   queue */
  MQCHAR48  ResolvedQMgrName;   /* Resolved name of destination queue
                                   manager */
  /* Ver:1 */
  MQLONG    RecsPresent;        /* Number of put message records or
                                   response records present */
  MQLONG    PutMsgRecFields;    /* Flags indicating which MQPMR fields
                                   are present */
  MQLONG    PutMsgRecOffset;    /* Offset of first put message record
                                   from start of MQPMO */
  MQLONG    ResponseRecOffset;  /* Offset of first response record
                                   from start of MQPMO */
  MQPTR     PutMsgRecPtr;       /* Address of first put message
                                   record */
  MQPTR     ResponseRecPtr;     /* Address of first response record */
  /* Ver:2 */
  MQHMSG    OriginalMsgHandle;  /* Original message handle */
  MQHMSG    NewMsgHandle;       /* New message handle */
  MQLONG    Action;             /* The action being performed */
  MQLONG    PubLevel;           /* Subscription level */
  /* Ver:3 */
};
```

Deklaracja COBOL dla MQPMO

```
**   MQPMO structure
  10 MQPMO.
**    Structure identifier
   15 MQPMO-STRUCID           PIC X(4).
**    Structure version number
   15 MQPMO-VERSION           PIC S9(9) BINARY.
**    Options that control the action of MQPUT and MQPUT1
   15 MQPMO-OPTIONS           PIC S9(9) BINARY.
**    Reserved
   15 MQPMO-TIMEOUT           PIC S9(9) BINARY.
**    Object handle of input queue
   15 MQPMO-CONTEXT           PIC S9(9) BINARY.
**    Number of messages sent successfully to local queues
   15 MQPMO-KNOWNDESTCOUNT    PIC S9(9) BINARY.
**    Number of messages sent successfully to remote queues
   15 MQPMO-UNKNOWNDESTCOUNT  PIC S9(9) BINARY.
**    Number of messages that could not be sent
   15 MQPMO-INVALIDDESTCOUNT  PIC S9(9) BINARY.
**    Resolved name of destination queue
   15 MQPMO-RESOLVEDQNAME     PIC X(48).
**    Resolved name of destination queue manager
   15 MQPMO-RESOLVEDQMGRNAME  PIC X(48).
**    Number of put message records or response records present
   15 MQPMO-RECSPRESENT       PIC S9(9) BINARY.
**    Flags indicating which MQPMR fields are present
   15 MQPMO-PUTMSGRECFIELDS   PIC S9(9) BINARY.
**    Offset of first put message record from start of MQPMO
   15 MQPMO-PUTMSGRECOFFSET   PIC S9(9) BINARY.
**    Offset of first response record from start of MQPMO
   15 MQPMO-RESPONSERECOFFSET PIC S9(9) BINARY.
**    Address of first put message record
   15 MQPMO-PUTMSGRECPTR      POINTER.
**    Address of first response record
   15 MQPMO-RESPONSERECPTR    POINTER.
**    Original message handle
   15 MQPMO-ORIGINALMSGHANDLE PIC S9(18) BINARY.
**    New message handle
   15 MQPMO-NEWMSGHANDLE      PIC S9(18) BINARY.
**    The action being performed
   15 MQPMO-ACTION            PIC S9(9) BINARY.
**    Publish level
   15 MQPMO-PUBLEVEL          PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQPMO

```
dcl
 1 MQPMO based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 Options           fixed bin(31), /* Options that control the action
```

```
                                       of MQPUT and MQPUT1 */
  3 Timeout           fixed bin(31), /* Reserved */
  3 Context           fixed bin(31), /* Object handle of input queue */
  3 KnownDestCount    fixed bin(31), /* Number of messages sent
                                       successfully to local queues */
  3 UnknownDestCount  fixed bin(31), /* Number of messages sent
                                       successfully to remote queues */
  3 InvalidDestCount  fixed bin(31), /* Number of messages that could
                                       not be sent */
  3 ResolvedQName     char(48),      /* Resolved name of destination
                                       queue */
  3 ResolvedQMgrName  char(48),      /* Resolved name of destination
                                       queue manager */
  3 RecsPresent       fixed bin(31), /* Number of put message records or
                                       response records present */
  3 PutMsgRecFields   fixed bin(31), /* Flags indicating which MQPMR
                                       fields are present */
  3 PutMsgRecOffset   fixed bin(31), /* Offset of first put message
                                       record from start of MQPMO */
  3 ResponseRecOffset fixed bin(31), /* Offset of first response record
                                       from start of MQPMO */
  3 PutMsgRecPtr      pointer,       /* Address of first put message
                                       record */
  3 ResponseRecPtr    pointer,       /* Address of first response
                                       record */
  3 OriginalMsgHandle fixed bin(63), /* Original message handle */
  3 NewMsgHandle      fixed bin(63); /* New message handle */
  3 Action            fixed bin(31); /* The action being performed */
  3 PubLevel          fixed bin(31); /* Publish level */
```

Deklaracja High Level Assembler dla MQPMO

```
MQPMO                   DSECT
MQPMO_STRUCID           DS   CL4   Structure identifier
MQPMO_VERSION           DS   F     Structure version number
MQPMO_OPTIONS           DS   F     Options that control the action of
*                                  MQPUT and MQPUT1
MQPMO_TIMEOUT           DS   F     Reserved
MQPMO_CONTEXT           DS   F     Object handle of input queue
MQPMO_KNOWNDESTCOUNT    DS   F     Number of messages sent successfully
*                                  to local queues
MQPMO_UNKNOWNDESTCOUNT  DS   F     Number of messages sent successfully
*                                  to remote queues
MQPMO_INVALIDDESTCOUNT  DS   F     Number of messages that could not be
*                                  sent
MQPMO_RESOLVEDQNAME     DS   CL48  Resolved name of destination queue
MQPMO_RESOLVEDQMGRNAME  DS   CL48  Resolved name of destination queue
*                                  manager
MQPMO_RECSPRESENT       DS   F     Number of put message records or
*                                  response records present
MQPMO_PUTMSGRECFIELDS   DS   F     Flags indicating which MQPMR
*                                  fields are present
MQPMO_PUTMSGRECOFFSET   DS   F     Offset of first put message record
*                                  from start of MQPMO
MQPMO_RESPONSERECOFFSET DS   F     Offset of first response record
*                                  from start of MQPMO
MQPMO_PUTMSGRECPTR      DS   F     Address of first put message
*                                  record
MQPMO_RESPONSERECPTR    DS   F     Address of first response record
MQPMO_ORIGINALMSGHANDLE DS   D     Original message handle
MQPMO_NEWMSGHANDLE      DS   D     New message handle
MQPMO_ACTION            DS   F     The action being performed
MQPMO_PUBLEVEL          DS   F     Publish level
*
MQPMO_LENGTH            EQU  *-MQPMO
                        ORG  MQPMO
MQPMO_AREA              DS   CL(MQPMO_LENGTH)
```

Deklaracja Visual Basic dla MQPMO

```
Type MQPMO
  StrucId           As String*4  'Structure identifier'
  Version           As Long      'Structure version number'
  Options           As Long      'Options that control the action of'
                                 'MQPUT and MQPUT1'
  Timeout           As Long      'Reserved'
  Context           As Long      'Object handle of input queue'
```

```
   KnownDestCount    As Long      'Number of messages sent successfully'
                                  'to local queues'
   UnknownDestCount  As Long      'Number of messages sent successfully'
                                  'to remote queues'
   InvalidDestCount  As Long      'Number of messages that could not be'
                                  'sent'
   ResolvedQName     As String*48 'Resolved name of destination queue'
   ResolvedQMgrName  As String*48 'Resolved name of destination queue'
                                  'manager'
   RecsPresent       As Long      'Number of put message records or'
                                  'response records present'
   PutMsgRecFields   As Long      'Flags indicating which MQPMR fields'
                                  'are present'
   PutMsgRecOffset   As Long      'Offset of first put message record'
                                  'from start of MQPMO'
   ResponseRecOffset As Long      'Offset of first response record from'
                                  'start of MQPMO'
   PutMsgRecPtr      As MQPTR     'Address of first put message record'
   ResponseRecPtr    As MQPTR     'Address of first response record'
 End Type
```

StrucId (MQCHAR4) dla MQPMO

Jest to identyfikator struktury struktury opcji umieszczania komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQPMO_STRUC_ID.

Wartość musi być następująca:

MQPMO_STRUC_ID,
Identyfikator struktury opcji umieszczania komunikatu.

Dla języka programowania C zdefiniowana jest również stała MQPMO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQPMO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQPMO

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

MQPMO_VERSION_1
Version-1 struktura opcji put-message.

Ta wersja jest obsługiwana we wszystkich środowiskach.

MQPMO_VERSION_2
Version-2 struktura opcji put-message.

Ta wersja jest obsługiwana w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

MQPMO_VERSION_3
Struktura opcji put-message Version-3 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQPMO_CURRENT_VERSION
Bieżąca wersja struktury opcji put-message.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQPMO_VERSION_1.

Opcje (MQLONG) dla MQPMO

Pole Opcje steruje działaniem wywołań **MQPUT** i **MQPUT1** .

Opcja zasięgu. Można podać dowolną lub żadną z opcji MQPMO. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe). Kombinacje, które nie są poprawne, są odnotowywane; wszystkie inne kombinacje są poprawne.

Następująca opcja steruje zasięgiem wysyłanych publikacji:

MQPMO_SCOPE_QMGR (menedżer kolejek)

Publikacja jest wysyłana tylko do subskrybentów, którzy zasubskrybowali w tym menedżerze kolejek. Publikacja nie jest przekazywana do żadnych zdalnych menedżerów kolejek publikowania/subskrypcji, które dokonały subskrypcji w tym menedżerze kolejek, co powoduje przesłonięcie zachowania ustawionego przy użyciu atrybutu tematu PUBSCOPE.

Uwaga: Jeśli nie jest ustawiony, zasięg publikacji jest określany przez atrybut tematu PUBSCOPE.

Opcje publikowania. Następujące opcje sterują sposobem publikowania komunikatów w temacie:

MQPMO_SUPPRESS_REPLYTO

Żadne informacje określone w polach *ReplyToQ* i *ReplyToQMgr* deskryptora MQMD tej publikacji nie są przekazywane do subskrybentów. Jeśli ta opcja jest używana z opcją raportu, która wymaga *ReplyToQ*, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_MISSING_REPLY_TO_Q.

MQPMO_RETAIN

Wysyłana publikacja ma zostać zachowana przez menedżer kolejek. Ten czas przechowywania umożliwia subskrybentowi żądanie kopii tej publikacji po jej opublikowaniu za pomocą wywołania MQSUBRQ. Umożliwia również wysyłanie publikacji do aplikacji, które mają swoją subskrypcję po utworzeniu publikacji (chyba że publikacja nie zostanie wysłana za pomocą opcji MQSO_NEW_PUBLIC). Jeśli do aplikacji zostanie wysłana zachowana publikacja, jest ona wskazywana przez właściwość komunikatu MQIsRetained tej publikacji.

W każdym węźle drzewa tematów może być zachowana tylko jedna publikacja. Dlatego też, jeśli istnieje już zachowana publikacja dla tego tematu, opublikowana przez dowolną inną aplikację, jest ona zastępowana tą publikacją. Dlatego lepiej jest unikać sytuacji, w której więcej niż jeden publikator zachowuje komunikaty w tym samym temacie.

Jeśli subskrybent żąda zachowanych publikacji, używana subskrypcja może zawierać znak wieloznaczny w temacie. W takim przypadku pewna liczba zachowanych publikacji może być zgodna (w różnych węzłach drzewa tematów) i kilka publikacji może zostać wysłanych do aplikacji żądającej. Więcej szczegółów zawiera opis wywołania “MQSUBRQ-żądanie subskrypcji” na stronie 821 .

Informacje na temat interakcji zachowanych publikacji z poziomami subskrypcji zawiera sekcja Przechwytywanie publikacji.

Jeśli ta opcja jest używana i nie można zachować publikacji, komunikat nie jest publikowany i wywołanie kończy się niepowodzeniem z wartością MQRC_PUT_NOT_ZACHOWANY.

MQPMO_NOT_OWN_SUBS,

Informuje menedżer kolejek, że aplikacja nie chce wysyłać żadnych swoich publikacji do subskrypcji, których jest właścicielem. Subskrypcje są uznawane za należące do tej samej aplikacji, jeśli uchwyty połączenia są takie same.

MQPMO_WARN_IF_NO_SUBS_MATCHED

Jeśli żadna subskrypcja nie jest zgodna z publikacją, zwróć kod zakończenia (*CompCode*) MQCC_WARNING i kod przyczyny MQRC_NO_SUBS_MATCHED.

Jeśli operacja put zwróci wartość MQRC_NO_SUBS_MATCHED, publikacja nie została dostarczona do żadnych subskrypcji. Jeśli jednak w operacji umieszczania określono opcję MQPMO_RETAIN, komunikat jest zachowywany i dostarczany do każdej następnej zgodnej subskrypcji.

Subskrypcja tematu jest zgodna z publikacją, jeśli spełniony jest dowolny z następujących warunków:

- Komunikat jest dostarczany do kolejki subskrypcji
- Komunikat zostałby dostarczony do kolejki subskrypcji, ale problem z kolejką oznacza, że komunikat nie może zostać umieszczony w kolejce i w związku z tym został umieszczony w kolejce niedostarczonych komunikatów lub odrzucony.
- Zdefiniowane jest wyjście routingu, które wyłącza dostarczanie komunikatu do subskrypcji.

Subskrypcja tematu nie jest zgodna z publikacją, jeśli spełniony jest dowolny z następujących warunków:

- Subskrypcja zawiera łańcuch wyboru, który nie jest zgodny z publikacją
- Subskrypcja określiła opcję MQSO_PUBLIC ON_REQUEST
- Publikacja nie została dostarczona, ponieważ w operacji umieszczania podano opcję MQPMO_NOT_OWN_SUBS, a subskrypcja jest zgodna z tożsamością publikatora.

Opcje punktu synchronizacji. Następujące opcje odnoszą się do udziału wywołania MQPUT lub MQPUT1 w jednostce pracy:

MQPMO_SYNCPOINT
Żądanie ma działać w ramach zwykłych protokołów jednostki pracy. Komunikat nie jest widoczny poza jednostką pracy, dopóki jednostka pracy nie zostanie zatwierdzona. Jeśli jednostka pracy zostanie wycofana, komunikat zostanie usunięty.

Jeśli nie określono opcji MQPMO_SYNCPOINT i MQPMO_NO_SYNCPOINT, włączenie żądania put do protokołów jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek, a nie przez środowisko, w którym działa aplikacja. W systemie z/OSżądanie umieszczenia znajduje się w jednostce pracy. We wszystkich innych środowiskach żądanie umieszczenia nie znajduje się w jednostce pracy.

Z powodu tych różnic aplikacja, która ma być używana jako port, nie może zezwalać na ustawienie tej opcji jako domyślnej. Należy jawnie określić opcję MQPMO_SYNCPOINT lub MQPMO_NO_SYNCPOINT.

Nie należy podawać opcji MQPMO_SYNCPOINT z opcją MQPMO_NO_SYNCPOINT.

MQPMO_NO_SYNCPOINT
Żądanie ma działać poza normalnymi protokołami jednostki pracy. Komunikat jest dostępny natychmiast i nie można go usunąć przez wycofanie jednostki pracy.

Jeśli nie określono opcji MQPMO_NO_SYNCPOINT i MQPMO_SYNCPOINT, włączenie żądania put do protokołów jednostki pracy jest określane przez środowisko, w którym działa menedżer kolejek, a nie przez środowisko, w którym działa aplikacja. W systemie z/OSżądanie umieszczenia znajduje się w jednostce pracy. We wszystkich innych środowiskach żądanie umieszczenia nie znajduje się w jednostce pracy.

Z powodu tych różnic aplikacja, która ma być używana jako port, nie może zezwalać na ustawienie tej opcji jako domyślnej. Należy jawnie określić opcję MQPMO_SYNCPOINT lub MQPMO_NO_SYNCPOINT.

Nie należy podawać opcji MQPMO_NO_SYNCPOINT z opcją MQPMO_SYNCPOINT.

Opcje Message-identifier i correlation-identifier. Następujące opcje żądają od menedżera kolejek wygenerowania nowego identyfikatora komunikatu lub identyfikatora korelacji:

MQPMO_NEW_MSG_ID
Menedżer kolejek zastępuje treść pola *MsgId* w strukturze MQMD nowym identyfikatorem komunikatu. Ten identyfikator komunikatu jest wysyłany wraz z komunikatem i zwracany do aplikacji w wyniku wywołania MQPUT lub MQPUT1 .

Opcję MQPMO_NEW_MSG_ID można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej. Szczegółowe informacje zawiera opis pola *MsgId* w strukturze MQPMR.

Użycie tej opcji zmniejsza konieczność zresetowania pola *MsgId* na wartość MQMI_NONE przed każdym wywołaniem MQPUT lub MQPUT1 .

MQPMO_NEW_CORREL_ID
Menedżer kolejek zastępuje treść pola *CorrelId* w strukturze MQMD nowym identyfikatorem korelacji. Ten identyfikator korelacji jest wysyłany z komunikatem i zwracany do aplikacji w wyniku wywołania MQPUT lub MQPUT1 .

Opcję MQPMO_NEW_CORREL_ID można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej. Szczegółowe informacje zawiera opis pola *CorrelId* w strukturze MQPMR.

Parametr MAQPMO_NEW_CORREL_ID jest przydatny w sytuacjach, w których aplikacja wymaga unikalnego identyfikatora korelacji.

Opcje grup i segmentów. Poniższe opcje odnoszą się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Zapoznaj się z poniższymi definicjami, aby zrozumieć tę opcję.

Ostrzeżenie: Z publikowaniem/subskrybowaniem nie można używać segmentowanych lub zgrupowanych komunikatów.

Komunikat fizyczny
Jest to najmniejsza jednostka informacji, którą można umieścić w kolejce lub z niej usunąć. Często odpowiada ona informacjom określonym lub pobranym w pojedynczym wywołaniu MQPUT, MQPUT1lub MQGET. Każdy komunikat fizyczny ma własny deskryptor komunikatu (MQMD). Zazwyczaj komunikaty fizyczne są rozróżniane na podstawie różnych wartości identyfikatora komunikatu (pole*MsgId* w strukturze MQMD), chociaż nie jest to wymuszane przez menedżer kolejek.

Komunikat logiczny
Komunikat logiczny jest pojedynczą jednostką informacji o aplikacji tylko dla platform innych niż z/OS . W przypadku braku ograniczeń systemowych komunikat logiczny jest taki sam jak komunikat fizyczny. Jeśli jednak komunikaty logiczne są bardzo duże, ograniczenia systemowe mogą spowodować, że wskazane lub konieczne będzie podzielenie komunikatu logicznego na dwa lub więcej komunikatów fizycznych, nazywanych *segmentami*.

Komunikat logiczny, który został podzielony na segmenty, składa się z dwóch lub większej liczby komunikatów fizycznych, które mają ten sam identyfikator grupy o wartości innej niż NULL (pole*GroupId* w strukturze MQMD) i ten sam numer kolejny komunikatu (pole*MsgSeqNumber* w strukturze MQMD). Segmenty są rozróżniane przez różne wartości przesunięcia segmentu (pole*Offset* w strukturze MQMD), które powoduje przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dozwolona przez aplikację wysyłającą, ma również identyfikator grupy o wartości innej niż NULL, chociaż w tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają identyfikator grupy o wartości NULL (MQGI_NONE), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy
Grupa komunikatów jest zestawem jednego lub większej liczby komunikatów logicznych, które mają ten sam identyfikator grupy o wartości innej niż NULL. Komunikaty logiczne w grupie są rozróżniane przez różne wartości numeru kolejnego komunikatu, który jest liczbą całkowitą z zakresu od 1 do *n*, gdzie *n* jest liczbą komunikatów logicznych w grupie. Jeśli jeden lub więcej komunikatów logicznych jest segmentowanych, w grupie istnieje więcej niż *n* komunikatów fizycznych.

MQPMO_LOGICAL_ORDER,
Ta opcja informuje menedżera kolejek, w jaki sposób aplikacja umieszcza komunikaty w grupach i segmentach komunikatów logicznych. Można go określić tylko w wywołaniu MQPUT. Nie jest on poprawny w wywołaniu MQPUT1 .

Jeśli zostanie określona opcja MQPMO_LOGICAL_ORDER, oznacza to, że aplikacja używa kolejnych wywołań MQPUT w następujących celach:

1. Umieszczenie segmentów w każdym komunikacie logicznym w kolejności rosnącego przesunięcia segmentu, począwszy od 0, bez przerw.

2. Umieszczenie wszystkich segmentów w jednym komunikacie logicznym przed umieszczeniem segmentów w następnym komunikacie logicznym.
3. Umieszczenie komunikatów logicznych w każdej grupie komunikatów w kolejności rosnących numerów kolejnych komunikatów, począwszy od 1, bez przerw. Numer kolejny komunikatu jest zwiększany automatycznie w produkcie IBM MQ.
4. Umieszczenie wszystkich komunikatów logicznych w jednej grupie komunikatów przed umieszczeniem komunikatów logicznych w następnej grupie komunikatów.

Szczegółowe informacje na temat parametru MQPMO_LOGICAL_ORDER zawiera sekcja Porządkowanie logiczne i fizyczne .

Opcje kontekstu. Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

MQPMO_NO_CONTEXT
Zarówno tożsamość, jak i kontekst źródłowy są ustawione tak, aby wskazywały brak kontekstu. Oznacza to, że pola kontekstu w strukturze MQMD są ustawione na:

- Odstępy dla pól znakowych
- Wartości puste dla pól bajtowych
- Zera dla pól liczbowych

MQPMO_DEFAULT_CONTEXT,
Komunikat ma mieć powiązane domyślne informacje o kontekście, zarówno dla tożsamości, jak i dla źródła. Menedżer kolejek ustawia pola kontekstu w deskryptorze komunikatu w następujący sposób:

Tabela 508. Domyślne wartości informacji o kontekście dla pól MQMD

Pole w strukturze MQMD	Użyta wartość
UserIdentifier	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie ustawiana jest wartość pusta.
AccountingToken	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie ustawiana jest wartość MQACT_NONE.
ApplIdentityData	Ustaw na wartość pustą.
PutApplType	Określana na podstawie środowiska.
PutApplName	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie ustawiana jest wartość pusta.
PutDate	Ustawiana jest data umieszczenia komunikatu.
PutTime	Ustaw czas umieszczenia komunikatu.
ApplOriginData	Ustaw na wartość pustą.

Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

Są to wartości domyślne i działania, jeśli nie określono opcji kontekstu.

MQPMO_PASS_IDENTITY_CONTEXT,
Komunikat ma mieć powiązane informacje o kontekście. Kontekst tożsamości jest pobierany z uchwytu kolejki określonego w polu *Context* . Informacje o kontekście źródłowym są generowane przez menedżer kolejek w taki sam sposób, jak dla MQPMO_DEFAULT_CONTEXT (wartości można znaleźć w poprzedniej tabeli). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

W przypadku wywołania MQPUT kolejka musi być otwarta z opcją MQOO_PASS_IDENTITY_CONTEXT (lub z opcją, która ją implikuje). W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją MQOO_PASS_IDENTITY_CONTEXT.

MQPMO_PASS_ALL_CONTEXT
Komunikat ma mieć powiązane informacje o kontekście. Kontekst jest pobierany z uchwytu kolejki określonego w polu *Context* . Więcej informacji na temat kontekstu komunikatu zawiera sekcja Sterowanie kontekstem.

W przypadku wywołania MQPUT kolejka musi być otwarta za pomocą opcji MQOO_PASS_ALL_CONTEXT (lub opcji, która ją implikuje). W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją MQOO_PASS_ALL_CONTEXT.

MQPMO_SET_IDENTITY_CONTEXT,
Komunikat ma mieć powiązane informacje o kontekście. Aplikacja określa kontekst tożsamości w strukturze MQMD. Informacje o kontekście źródłowym są generowane przez menedżer kolejek w taki sam sposób, jak dla MQPMO_DEFAULT_CONTEXT (wartości można znaleźć w poprzedniej tabeli). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

W przypadku wywołania MQPUT kolejka musi być otwarta za pomocą opcji MQOO_SET_IDENTITY_CONTEXT (lub opcji, która ją implikuje). Dla wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co dla wywołania MQOPEN z opcją MQOO_SET_IDENTITY_CONTEXT.

MQPMO_SET_ALL_CONTEXT
Komunikat ma mieć powiązane informacje o kontekście. Aplikacja określa tożsamość, pochodzenie i kontekst użytkownika w strukturze MQMD. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

W przypadku wywołania MQPUT kolejka musi być otwarta z opcją MQOO_SET_ALL_CONTEXT. W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją MQOO_SET_ALL_CONTEXT.

Można podać tylko jedną z opcji kontekstu MQPMO_ * _CONTEXT. Jeśli nie zostanie podana żadna wartość, przyjmowana jest wartość MQPMO_DEFAULT_CONTEXT.

Opcje właściwości. Następująca opcja odnosi się do właściwości komunikatu:

MQPMO_MD_FOR_OUTPUT_ONLY,
Parametr deskryptora komunikatu może być używany tylko dla danych wyjściowych w celu zwrócenia deskryptora komunikatu, który został umieszczony. Pola deskryptora komunikatu powiązane z polami *NewMsgHandle*, *OriginalMsgHandle*lub obydwoma tymi polami struktury **MQPMO** muszą być używane jako dane wejściowe.

Jeśli nie podano poprawnego uchwytu komunikatu, wywołanie kończy się niepowodzeniem z kodem przyczyny **MQRC_MD_ERROR**.

Opcje umieszczania odpowiedzi. Następujące opcje sterują odpowiedzią zwracaną do wywołania MQPUT lub MQPUT1 . Można określić tylko jedną z tych opcji. Jeśli nie określono opcji MQPMO_ASYNC_RESPONSE i MQPMO_SYNC_RESPONSE, przyjmuje się opcję MQPMO_RESPONSE_AS_Q_DEF lub MQPMO_RESPONSE_AS_TOPIC_DEF.

MQPMO_ASYNC_RESPONSE
Opcja MQPMO_ASYNC_RESPONSE żąda zakończenia operacji MQPUT lub MQPUT1 bez oczekiwania aplikacji na zakończenie wywołania przez menedżer kolejek. Użycie tej opcji może zwiększyć wydajność przesyłania komunikatów, szczególnie w przypadku aplikacji korzystających z powiązań klienta. Aplikacja może okresowo sprawdzać za pomocą komendy MQSTAT, czy podczas poprzednich wywołań asynchronicznych wystąpił błąd.

W przypadku tej opcji w strukturze MQMD muszą być wypełnione tylko następujące pola:

- Dane_tożsamości_aplikacji
- Typ_aplikacji_wstawiającej
- Nazwa_aplikacji_wstawiającej
- Dane_pochodzenia_aplikacji

Dodatkowo, jeśli jako opcje podano jeden lub oba parametry MQPMO_NEW_MSG_ID lub MQPMO_NEW_CORREL_ID, zwracane są również wartości MsgId i CorrelId . (MQPMO_NEW_MSG_ID można określić niejawnie, określając puste pole MsgId).

Zostaną wypełnione tylko te pola, które zostały podane wcześniej. Inne informacje, które normalnie byłyby zwracane w strukturze MQMD lub MQPMO, są niezdefiniowane.

Podczas żądania asynchronicznej odpowiedzi put dla MQPUT1nazwy ResolvedQName i ResolvedQMgrzwracane w strukturze MQOD są niezdefiniowane.

W przypadku żądania asynchronicznej odpowiedzi umieszczania dla MQPUT lub MQPUT1wartość CompCode i Reason of MQCC_OK oraz MQRC_NONE nie musi oznaczać, że komunikat został pomyślnie umieszczony w kolejce. Podczas tworzenia aplikacji MQI używającej asynchronicznej odpowiedzi put i wymagającej potwierdzenia, że komunikaty zostały umieszczone w kolejce, należy sprawdzić zarówno kod CompCode , jak i kod przyczyny operacji put, a także użyć komendy MQSTAT do wysłania zapytania o asynchroniczne informacje o błędzie.

Mimo że powodzenie lub niepowodzenie każdego pojedynczego wywołania MQPUT lub MQPUT1 może nie zostać natychmiast zwrócone, pierwszy błąd, który wystąpił w wywołaniu asynchronicznym, można określić później za pomocą wywołania MQSTAT.

Jeśli dostarczenie komunikatu trwałego w punkcie synchronizacji nie powiedzie się przy użyciu asynchronicznej odpowiedzi umieszczania, a użytkownik podejmie próbę zatwierdzenia transakcji, zatwierdzenie nie powiedzie się, a transakcja zostanie wycofana z kodem zakończenia MQCC_FAILED i przyczyną MQRC_BACKED_OUT. Aplikacja może wywołać MQSTAT w celu określenia przyczyny poprzedniego niepowodzenia MQPUT lub MQPUT1 .

MQPMO_SYNC_RESPONSE
Określenie tego typu odpowiedzi umieszczania zapewnia, że operacja MQPUT lub MQPUT1 jest zawsze wykonywana synchronicznie. Jeśli operacja umieszczania powiedzie się, wszystkie pola w MQMD i MQPMO zostaną zakończone.

Ta opcja zapewnia odpowiedź synchroniczną bez względu na domyślną wartość odpowiedzi umieszczania zdefiniowaną w kolejce lub obiekcie tematu.

MQPMO_ODPOWIEDŹ_AS_Q_DEF
Jeśli ta wartość jest określona dla wywołania MQPUT, używany typ odpowiedzi umieszczania jest pobierany z wartości DEFPRESP określonej w kolejce podczas pierwszego otwarcia przez aplikację.

- Jeśli kolejka jest kolejką klastra, a ta wartość jest określona dla wywołania MQPUT, używany typ odpowiedzi umieszczania jest pobierany z atrybutu **DEFPRESP** zdefiniowanego w menedżerze kolejek *miejsca docelowego* , który jest właścicielem konkretnej instancji kolejki, w której umieszczany jest komunikat.

  Jeśli istnieje wiele instancji kolejki klastra różniących się tym atrybutem, pobierana jest wartość jednego z nich, ale nie można przewidzieć, która z tych wartości zostanie użyta. Dlatego też ten atrybut należy ustawić na tę samą wartość we wszystkich instancjach. Jeśli to nie jest przyczyna problemu, do dzienników menedżera kolejek wysyłany jest komunikat o błędzie AMQ9407. Należy również zapoznać się z sekcją Jak atrybuty obiektów docelowych są rozstrzygane w przypadku kolejek aliasowych, kolejek zdalnych i kolejek klastra?
- Jeśli kolejka nie jest kolejką klastra, a ta wartość jest określona dla wywołania MQPUT, używany typ odpowiedzi umieszczania jest pobierany z atrybutu **DEFPRESP** zdefiniowanego w *lokalnym* menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

Jeśli aplikacja kliencka jest połączona z menedżerem kolejek na poziomie wcześniejszym niż IBM WebSphere MQ 7.0, zachowuje się tak, jakby określono parametr MQPMO_SYNC_RESPONSE.

Jeśli ta opcja jest określona dla wywołania MQPUT1 , wartość atrybutu DEFPRESP nie jest znana przed wysłaniem żądania do serwera. Domyślnie, jeśli wywołanie MQPUT1 używa komendy MQPMO_SYNCPOINT, zachowuje się tak samo, jak w przypadku komendy MQPMO_ASYNC_RESPONSE, a jeśli używa wywołania MQPMO_NO_SYNCPOINT, zachowuje się tak, jak w przypadku komendy MQPMO_SYNC_RESPONSE. Można jednak przesłonić to zachowanie domyślne, ustawiając właściwość Put1DefaultAlwaysSync w pliku konfiguracyjnym klienta, patrz sekcja Sekcja CHANNELS w pliku konfiguracyjnym klienta.

MQPMO_ODPOWIEDŹ_AS_TOPIC_DEF
MQPMO_RESPONSE_AS_TOPIC_DEF jest synonimem MQPMO_RESPONSE_AS_Q_DEF do użycia z obiektami tematów.

Inne opcje. Następujące opcje sterują sprawdzaniem autoryzacji, sprawdzaniem, co się dzieje, gdy menedżer kolejek jest wyciszany oraz rozstrzyganiem nazw kolejek i menedżerów kolejek:

MQPMO_ALTERNATE_UPRAWNIENIA_UŻYTKOWNIKA
MQPMO_ALTERNATE_USER_AUTHORITY wskazuje, że pole *AlternateUserId* w parametrze **ObjDesc** wywołania MQPUT1 zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności uprawnień do umieszczania komunikatów w kolejce. Wywołanie może zakończyć się pomyślnie tylko wtedy, gdy program *AlternateUserId* ma uprawnienia do otwarcia kolejki z podanymi opcjami, bez względu na to, czy identyfikator użytkownika, pod którym działa aplikacja, jest do tego uprawniony. (Nie ma to jednak zastosowania do określonych opcji kontekstu, które są zawsze sprawdzane względem identyfikatora użytkownika, pod którym działa aplikacja).

Ta opcja jest poprawna tylko w połączeniu z wywołaniem MQPUT1 .

MQPMO_FAIL_IF_QUIESCING,
Ta opcja wymusza niepowodzenie wywołania MQPUT lub MQPUT1 , jeśli menedżer kolejek jest w stanie wyciszania.

W systemie z/OSta opcja wymusza również niepowodzenie wywołania MQPUT lub MQPUT1 , jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wyciszania.

Wywołanie zwraca kod zakończenia MQCC_FAILED z kodem przyczyny MQRC_Q_MGR_QUIESCING lub MQRC_CONNECTION_QUIESCING.

MQPMO_RESOLVE_LOCAL_Q,
Ta opcja służy do wypełniania pola *ResolvedQName* w strukturze MQPMO nazwą kolejki lokalnej, w której jest umieszczany komunikat, oraz pola *ResolvedQMgrName* nazwą menedżera kolejek lokalnych, który udostępnia kolejkę lokalną. Więcej informacji na temat komendy MQPMO_RESOLVE_LOCAL_Q zawiera temat MQOO_RESOLVE_LOCAL_Q.

Jeśli użytkownik ma uprawnienia do umieszczania w kolejce, ma uprawnienia wymagane do określenia tej flagi w wywołaniu MQPUT. Uprawnienia specjalne nie są wymagane.

Opcja domyślna. Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

MQPMO_BRAK
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. Parametr MQPMO_NONE jest zdefiniowany w celu wspomagania dokumentacji programu; nie jest on przeznaczony do użycia z żadną inną opcją, ale ponieważ jego wartość wynosi zero, użycie tej opcji nie może zostać wykryte.

MQPMO_NONE jest polem wejściowym. Wartością początkową pola *Options* jest MQPMO_NONE.

Limit czasu (MQLONG) dla MQPMO

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest -1.

Kontekst (MQHOBJ) dla MQPMO

Jeśli określono opcję MQPMO_PASS_IDENTITY_CONTEXT lub MQPMO_PASS_ALL_CONTEXT, to pole musi zawierać uchwyt kolejki wejściowej, z którego pobierane są informacje o kontekście powiązane z umieszczanym komunikatem.

Jeśli nie określono opcji MQPMO_PASS_IDENTITY_CONTEXT ani opcji MQPMO_PASS_ALL_CONTEXT, to pole jest ignorowane.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

KnownDest-liczba (MQLONG) dla MQPMO

Jest to liczba komunikatów, które zostały pomyślnie wysłane przez bieżące wywołania MQPUT lub MQPUT1 do kolejek znajdujących się na liście dystrybucyjnej, które są kolejkami lokalnymi. Licznik

nie uwzględnia komunikatów wysłanych do kolejek, które są tłumaczone na kolejki zdalne (nawet jeśli początkowo do przechowywania komunikatu używana jest lokalna kolejka transmisji). To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawiane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_1.

Ta zmienna jest niezdefiniowana w systemie z/OS , ponieważ listy dystrybucyjne nie są obsługiwane.

UnknownDestLiczba (MQLONG) dla MQPMO

Jest to liczba komunikatów pomyślnie wysłanych przez bieżące wywołanie MQPUT lub MQPUT1 do kolejek znajdujących się na liście dystrybucyjnej, które są tłumaczone na kolejki zdalne. Komunikaty przechowywane tymczasowo przez menedżera kolejek w postaci listy dystrybucyjnej są liczone jako liczba pojedynczych miejsc docelowych, które zawierają te listy dystrybucyjne. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawiane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_1.

Ta zmienna jest niezdefiniowana w systemie z/OS , ponieważ listy dystrybucyjne nie są obsługiwane.

Liczba operacji InvalidDest(MQLONG) dla MQPMO

Jest to liczba komunikatów, których nie można było wysłać do kolejek na liście dystrybucyjnej. Liczba ta obejmuje kolejki, których otwarcie nie powiodło się, a także kolejki, które zostały otwarte pomyślnie, ale dla których operacja umieszczania nie powiodła się. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Uwaga: To pole jest ustawiane, jeśli parametr **CompCode** w wywołaniu MQPUT lub MQPUT1 ma wartość MQCC_OK lub MQCC_WARNING; może być ustawiane, jeśli parametr **CompCode** ma wartość MQCC_FAILED, ale nie jest zależne od kodu aplikacji.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawiane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_1.

Ta zmienna jest niezdefiniowana w systemie z/OS , ponieważ listy dystrybucyjne nie są obsługiwane.

ResolvedQName (MQCHAR48) dla obiektu MQPMO

Jest to nazwa kolejki docelowej po wykonaniu tłumaczenia nazw przez menedżera kolejek lokalnych. Zwracana nazwa jest nazwą kolejki, która istnieje w menedżerze kolejek identyfikowanym przez *ResolvedQMgrName*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

Jest to pole wyjściowe. Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ResolvedQMgrNazwa (MQCHAR48) dla MQPMO

Jest to nazwa docelowego menedżera kolejek po wykonaniu tłumaczenia nazw przez lokalny menedżer kolejek. Zwracana nazwa jest nazwą menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *ResolvedQName*, i może być nazwą lokalnego menedżera kolejek.

Jeśli *ResolvedQName* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejek, do której należy lokalny menedżer kolejek, *ResolvedQMgrName* jest nazwą grupy współużytkowania kolejek. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, parametr *ResolvedQName* może być nazwą grupy współużytkowania kolejek lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejek (rodzaj zwracanej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

Jest to pole wyjściowe. Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

RecsPresent (MQLONG) dla MQPMO

Jest to liczba rekordów komunikatów umieszczenia MQPMR lub rekordów odpowiedzi MQRR, które zostały udostępnione przez aplikację. Ta liczba może być większa od zera tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Rekordy umieszczania komunikatów i rekordy odpowiedzi są opcjonalne; aplikacja nie musi udostępniać żadnych rekordów lub może udostępniać tylko rekordy jednego typu. Jeśli jednak aplikacja udostępnia rekordy obu typów, musi udostępniać rekordy *RecsPresent* każdego typu.

Wartość *RecsPresent* nie musi być taka sama, jak liczba miejsc docelowych na liście dystrybucyjnej. Jeśli podano zbyt wiele rekordów, nadwyżka nie jest używana. Jeśli podano zbyt mało rekordów, używane są wartości domyślne dla właściwości komunikatu dla miejsc docelowych, które nie mają rekordów umieszczania komunikatów (patrz sekcja *PutMsgRecOffset*).

Jeśli wartość parametru *RecsPresent* jest mniejsza od zera lub większa od zera, ale komunikat nie jest umieszczany na liście dystrybucyjnej, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RECS_PRESENT_ERROR.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.

PutMsgRecFields (MQLONG) dla MQPMO

To pole zawiera flagi wskazujące, które pola MQPMR są obecne w rekordach komunikatów umieszczania udostępnianych przez aplikację. Parametru *PutMsgRecFields* należy używać tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero lub oba parametry *PutMsgRecOffset* i *PutMsgRecPtr* mają wartość zero.

W przypadku pól, które są obecne, menedżer kolejek używa dla każdego miejsca docelowego wartości z pól w odpowiednim rekordzie komunikatu umieszczania. W przypadku pól, które nie są dostępne, menedżer kolejek używa wartości ze struktury MQMD.

Użyj co najmniej jednej z następujących opcji, aby wskazać, które pola są obecne w rekordach komunikatu umieszczania:

MQPMRF_ID_komunikatu
Pole identyfikatora komunikatu jest obecne.

MQPMRF_CORREL_ID (Identyfikator relacji MQ)
Pole identyfikatora korelacji jest obecne.

MQPMRF_GROUP_ID (identyfikator grupy MQPMRF_ID)
Pole identyfikatora grupy jest obecne.

MQPMRF_FEEDBACK
Pole informacji zwrotnej jest obecne.

MQPMRF_ACCOUNTING_TOKEN,
Rozliczanie-pole tokenu jest obecne.

Jeśli ta opcja zostanie określona, w polu *Options* należy podać wartość MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT. Jeśli ten warunek nie zostanie spełniony, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_PMO_RECORD_FLAGS_ERROR.

Jeśli nie ma pól MQPMR, można podać następujące informacje:

MQPMRF_BRAK
Brak pól rekordu komunikatu umieszczania (put-message).

Jeśli ta wartość jest określona, parametr *RecsPresent* musi mieć wartość zero lub oba parametry *PutMsgRecOffset* i *PutMsgRecPtr* muszą mieć wartość zero.

Parametr MQPMRF_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta stała była używana z żadną inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Jeśli plik *PutMsgRecFields* zawiera niepoprawne flagi lub jeśli podano rekordy komunikatu umieszczania, ale parametr *PutMsgRecFields* ma wartość MQPMRF_NONE, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_PMO_RECORD_FLAGS_ERROR.

Jest to pole wejściowe. Wartością początkową tego pola jest MQPMRF_NONE. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.

PutMsgRecOffset (MQLONG) dla MQPMO

Jest to przesunięcie w bajtach pierwszego rekordu komunikatu umieszczonego przez MQPMR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. Parametr *PutMsgRecOffset* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Gdy komunikat jest umieszczany na liście dystrybucyjnej, można podać tablicę zawierającą jeden lub więcej rekordów komunikatów MQPMR typu put w celu określenia pewnych właściwości komunikatu dla każdego miejsca docelowego osobno. Są to następujące właściwości:

- Identyfikator komunikatu
- Identyfikator korelacji
- Identyfikator grupy
- Wartość opinii
- Token rozliczania

Nie ma potrzeby określania wszystkich tych właściwości, ale niezależnie od wybranego podzbioru należy określić pola w poprawnej kolejności. Więcej informacji zawiera opis struktury MQPMR.

Zwykle podczas otwierania listy dystrybucyjnej musi istnieć tyle rekordów komunikatów umieszczania, ile jest rekordów obiektów określonych przez program MQOD. Każdy rekord komunikatów umieszczania dostarcza właściwości komunikatu dla kolejki identyfikowanej przez odpowiedni rekord obiektu. Kolejki na liście dystrybucyjnej, których otwarcie nie powiodło się, muszą mieć przydzielone rekordy komunikatów na odpowiednich pozycjach w tablicy, chociaż w tym przypadku właściwości komunikatu są ignorowane.

Liczba rekordów komunikatów umieszczania może różnić się od liczby rekordów obiektów. Jeśli istnieje mniej rekordów komunikatów umieszczania niż rekordów obiektów, właściwości komunikatów dla miejsc docelowych, które nie mają rekordów komunikatów umieszczania, są pobierane z odpowiednich pól deskryptora komunikatu MQMD. Jeśli istnieje więcej rekordów komunikatów umieszczania niż rekordów obiektów, nadmiar nie jest używany (chociaż dostęp do nich musi być nadal możliwy). Rekordy umieszczania komunikatów są opcjonalne, ale jeśli zostaną podane, musi być ich *RecsPresent* .

Udostępnij rekordy umieszczania komunikatów w podobny sposób, jak rekordy obiektów w MQOD, określając przesunięcie w *PutMsgRecOffset*lub określając adres w *PutMsgRecPtr* ; Szczegółowe informacje na ten temat zawiera opis pola *ObjectRecOffset* w sekcji"MQOD-deskryptor obiektu" na stronie 493.

Nie można użyć więcej niż jednej z wartości *PutMsgRecOffset* i *PutMsgRecPtr* . Wywołanie nie powiedzie się z kodem przyczyny MQRC_PUT_MSG_RECORDS_ERROR, jeśli obie wartości są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.

ResponseRecprzesunięcie (MQLONG) dla MQPMO

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. Parametr *ResponseRecOffset* jest używany tylko

wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Podczas umieszczania komunikatu na liście dystrybucyjnej można udostępnić tablicę zawierającą jeden lub więcej rekordów odpowiedzi MQRR w celu zidentyfikowania kolejek, do których komunikat nie został pomyślnie wysłany (pole*CompCode* w MQRR), oraz przyczyny każdego niepowodzenia (pole*Reason* w MQRR). Komunikat mógł nie zostać wysłany, ponieważ otwarcie kolejki nie powiodło się lub operacja umieszczania nie powiodła się. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (to znaczy niektóre komunikaty zostały wysłane pomyślnie, podczas gdy inne nie powiodły się lub wszystkie nie powiodły się, ale z różnych przyczyn). Kod przyczyny MQRC_MULTIPLE_REASON z wywołania wskazuje ten przypadek. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna jest zwracana w parametrze **Reason** wywołania MQPUT lub MQPUT1 , a rekordy odpowiedzi nie są ustawiane.

Zwykle istnieje tyle rekordów odpowiedzi, ile jest rekordów obiektów określonych przez program MQOD podczas otwierania listy dystrybucyjnej; w razie potrzeby każdy rekord odpowiedzi jest ustawiany na kod zakończenia i kod przyczyny dla operacji umieszczania w kolejce identyfikowanej przez odpowiedni rekord obiektu. Kolejki na liście dystrybucyjnej, których otwarcie nie powiodło się, muszą nadal mieć przydzielone rekordy odpowiedzi na odpowiednich pozycjach w tablicy, mimo że są one ustawione na kod zakończenia i kod przyczyny wynikające z operacji otwierania, a nie operacji umieszczania (put).

Liczba rekordów odpowiedzi może różnić się od liczby rekordów obiektów. Jeśli liczba rekordów odpowiedzi jest mniejsza niż liczba rekordów obiektów, aplikacja może nie być w stanie zidentyfikować wszystkich miejsc docelowych, dla których operacja umieszczania nie powiodła się, lub przyczyn niepowodzeń. Jeśli liczba rekordów odpowiedzi jest większa niż liczba rekordów obiektów, nadwyżka nie jest używana (chociaż dostęp do niej musi być nadal możliwy). Rekordy odpowiedzi są opcjonalne, ale jeśli zostaną podane, musi być ich *RecsPresent* .

Udostępnij rekordy odpowiedzi w podobny sposób, jak rekordy obiektów w programie MQOD, określając przesunięcie w programie *ResponseRecOffset*lub określając adres w programie *ResponseRecPtr* . Szczegółowe informacje na ten temat zawiera opis pola *ObjectRecOffset* w sekcji “MQOD-deskryptor obiektu” na stronie 493. Należy jednak używać nie więcej niż jednego z następujących produktów: *ResponseRecOffset* i *ResponseRecPtr* ; Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_RESPONSE_RECORDS_ERROR, jeśli oba są niezerowe.

W przypadku wywołania MQPUT1 to pole musi mieć wartość zero. Dzieje się tak, ponieważ informacje o odpowiedzi (jeśli są żądane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.

PutMsgRecPtr (MQPTR) dla MQPMO

Jest to adres pierwszego rekordu umieszczenia komunikatu MQPMR. Parametru *PutMsgRecPtr* należy używać tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Do określenia rekordów umieszczania komunikatów można użyć wartości *PutMsgRecPtr* lub *PutMsgRecOffset* , ale nie obu tych wartości. Szczegółowe informacje na ten temat zawiera sekcja “PutMsgRecOffset (MQLONG) dla MQPMO” na stronie 529. Jeśli nie jest używany parametr *PutMsgRecPtr*, należy ustawić go na wskaźnik pusty lub bajty puste.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, to pole jest deklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null.

ResponseRecPtr (MQPTR) dla MQPMO

Jest to adres pierwszego rekordu odpowiedzi MQRR. Parametr *ResponseRecPtr* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *RecsPresent* ma wartość zero.

Do określenia rekordów odpowiedzi należy użyć wartości *ResponseRecPtr* lub *ResponseRecOffset* , ale nie obu tych wartości. Szczegółowe informacje na ten temat zawiera sekcja “ResponseRecprzesunięcie (MQLONG) dla MQPMO” na stronie 529. Jeśli nie jest używany parametr *ResponseRecPtr* , należy ustawić go na wskaźnik pusty lub bajty puste.

W przypadku wywołania MQPUT1 to pole musi być wskaźnikiem pustym lub bajtami o wartości NULL. Dzieje się tak, ponieważ informacje o odpowiedzi (jeśli są żądane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż wartość MQPMO_VERSION_2.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, to pole jest deklarowane jako łańcuch bajtowy o odpowiedniej długości, przy czym wartością początkową jest łańcuch bajtowy o wartości all-null.

OriginalMsgUchwyt (MQHMSG) dla MQPMO

Jest to opcjonalny uchwyt komunikatu. Możliwe, że został on wcześniej pobrany z kolejki. Użycie tego uchwytu zależy od wartości pola *Action* . Patrz także: NewMsgHandle.

Treść oryginalnego uchwytu komunikatu nie zostanie zmieniona przez wywołanie metody **MQPUT** lub **MQPUT1** .

Jest to pole wejściowe. Wartością początkową tego pola jest **MQHM_NONE**. To pole jest ignorowane, jeśli wartość w polu Wersja jest mniejsza niż **MQPMO_VERSION_3**.

NewMsgUchwyt (MQHMSG) dla MQPMO

Jest to opcjonalny uchwyt komunikatu umieszczanego w zależności od wartości pola Działanie. Definiuje on właściwości komunikatu i nadpisuje wartości *OriginalMsgHandle*, jeśli zostały określone.

Po powrocie z wywołania **MQPUT** lub **MQPUT1** treść uchwytu odzwierciedla komunikat, który został rzeczywiście umieszczony.

Jest to pole wejściowe. Wartością początkową tego pola jest **MQHM_NONE**. To pole jest ignorowane, jeśli wartość w polu Wersja jest mniejsza niż **MQPMO_VERSION_3**.

Działanie (MQLONG) dla MQPMO

Określa typ wykonywanego umieszczania oraz relację między oryginalnym komunikatem określonym w polu uchwytu OriginalMsgi nowym komunikatem określonym w polu uchwytu NewMsg. Właściwości komunikatu są wybierane przez menedżer kolejek zgodnie z określoną wartością działania.

Treść deskryptora komunikatu można podać za pomocą parametru MsgDesc w wywołaniach MQPUT lub MQPUT1 . Alternatywnie można nie podawać parametru MsgDesc lub określić, że jest on przeznaczony tylko dla danych wyjściowych, dołączając opcję MQPMO_MD_FOR_OUTPUT_ONLY do pola opcji struktury MQPMO.

Jeśli nie podano parametru MsgDesc lub określono go jako parametr tylko wyjściowy, deskryptor nowego komunikatu jest zapełniany na podstawie pól uchwytu komunikatu MQPMO zgodnie z regułami opisanymi w tym temacie.

Ustawienia kontekstu i działania przekazywania opisane w sekcji Kontrolowanie informacji o kontekście są uwzględniane po utworzeniu deskryptora komunikatu.

Jeśli podano niepoprawną wartość działania, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ACTION_ERROR.

Można określić jedno z następujących działań:

MQACTP_NOWA
Umieszczany jest nowy komunikat i program nie określa żadnego związku z poprzednim komunikatem. Deskryptor komunikatu składa się z następujących elementów:

- Jeśli parametr MsgDesc jest podany w wywołaniu MQPUT lub MQPUT1 , a parametr MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w MQPMO.Optionsjest używana jako niezmodyfikowany deskryptor komunikatu.
- Jeśli parametr MsgDesc nie został podany lub jeśli parametr MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w produkcie MQPMO.Options powoduje, że menedżer kolejek generuje deskryptor komunikatu przy użyciu kombinacji właściwości z uchwytu OriginalMsgi uchwytu NewMsg. Wszystkie pola deskryptora komunikatu jawnie ustawione w nowym uchwycie komunikatu mają pierwszeństwo przed polami w oryginalnym uchwycie komunikatu.

Dane komunikatu są pobierane z parametru buforu MQPUT lub MQPUT1 .

MQACTP_FORWARD
Poprzednio pobrany komunikat jest przekazywany. Oryginalny uchwyt komunikatu określa komunikat, który został wcześniej pobrany.

Nowy uchwyt komunikatu określa wszelkie modyfikacje właściwości (w tym dowolnej w deskryptorze komunikatu) w oryginalnym uchwycie komunikatu.

Deskryptor komunikatu składa się z następujących elementów:

- Jeśli parametr MsgDesc jest podany w wywołaniu MQPUT lub MQPUT1 , a parametr MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w MQPMO.Optionsjest używana jako niezmodyfikowany deskryptor komunikatu.
- Jeśli parametr MsgDesc nie został podany lub jeśli parametr MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w produkcie MQPMO.Options powoduje, że menedżer kolejek generuje deskryptor komunikatu przy użyciu kombinacji właściwości z uchwytu OriginalMsgi uchwytu NewMsg. Wszystkie pola deskryptora komunikatu jawnie ustawione w nowym uchwycie komunikatu mają pierwszeństwo przed polami w oryginalnym uchwycie komunikatu.
- Jeśli w parametrze MQPMO.Options, to są honorowane.

Właściwości komunikatu są tworzone w następujący sposób:

- Wszystkie właściwości z oryginalnego uchwytu komunikatu, które mają opcję MQCOPY_FORWARD w produkcie MQPD.CopyOptions
- Wszystkie właściwości z nowego uchwytu komunikatu. Dla każdej właściwości w nowym uchwycie komunikatu, która ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, wartość jest pobierana z nowego uchwytu komunikatu. Jedynym wyjątkiem od tej reguły jest przypadek specjalny, gdy właściwość w nowym uchwycie komunikatu ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, ale właściwość ma wartość NULL. W takim przypadku właściwość jest usuwana z komunikatu.

Dane komunikatu do przekazania są pobierane z parametru buforu MQPUT lub MQPUT1 .

ODPOWIEDŹ MQACTP_REPLY
Odpowiedź jest tworzona na poprzednio pobrany komunikat. Oryginalny uchwyt komunikatu określa komunikat, który został wcześniej pobrany.

Nowy uchwyt komunikatu określa wszelkie modyfikacje właściwości (w tym dowolnej w deskryptorze komunikatu) w oryginalnym uchwycie komunikatu.

Deskryptor komunikatu składa się z następujących elementów:

- Jeśli parametr MsgDesc jest podany w wywołaniu MQPUT lub MQPUT1 , a parametr MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w MQPMO.Optionsjest używana jako niezmodyfikowany deskryptor komunikatu.

- Jeśli parametr MsgDesc nie został podany lub jeśli parametr MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w produkcie MQPMO.Options, a następnie początkowe pola deskryptora komunikatu są wybierane w następujący sposób:

Tabela 509. Transformacja uchwytu komunikatu odpowiedzi

Pole w strukturze MQMD	**Użyta wartość**
Raport	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI i MQRO_DISCARD_MSG są ustawione: MQRO_DISCARD_MSG w przeciwnym razie MQRO_BRAK
MsgType	MQMT_REPLY
Utrata ważności	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI jest ustawione: Skopiowane z komunikatu wejściowego w przeciwnym razie MQEI_UNLIMITED,
Opinie	MQFB_BRAK
MsgId	Jeśli ustawiono MQPMO_NEW_MSG_ID: Generowany jest nowy identyfikator komunikatu w przeciwnym razie, jeśli ustawiono MQRO_PASS_MSG_ID: Skopiowane z komunikatu wejściowego w przeciwnym razie MQMI_BRAK
CorrelId	Jeśli ustawiono wartość MQPMO_NEW_CORREL_ID: Zostanie wygenerowany nowy identyfikator korelacji w przeciwnym razie, jeśli ustawiono MQRO_COPY_MSG_ID_TO_CORREL_ID: Skopiowany z pola MsgId w Komunikat wejściowy w przeciwnym razie, jeśli ustawiono MQRO_PASS_CORREL_ID: Skopiowane z pola CorrelId Komunikat wejściowy w przeciwnym razie MQCI_BRAK
BackoutCount	0
ReplyToQ	Puste
ReplyToQMgr	Puste
GroupId	MQGI_NONE
Numer_kolejny_komunikatu	1
Depozycja	0
MsgFlags	MQMF_BRAK
OriginalLength	MQOL_UNDEFINED

- Deskryptor komunikatu jest następnie modyfikowany przez nowy uchwyt komunikatu-wszystkie pola deskryptora komunikatu jawnie ustawione jako właściwości w nowym uchwycie komunikatu mają pierwszeństwo przed polami deskryptora komunikatu zgodnie z wcześniejszym opisem.

Właściwości komunikatu są tworzone w następujący sposób:

- Wszystkie właściwości z oryginalnego uchwytu komunikatu, które mają wartość MQCOPY_REPLY w MQPD.CopyOptions
- Wszystkie właściwości z nowego uchwytu komunikatu. Dla każdej właściwości w nowym uchwycie komunikatu, która ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, wartość jest pobierana z nowego uchwytu komunikatu. Jedynym wyjątkiem od tej reguły jest przypadek specjalny, gdy właściwość w nowym uchwycie komunikatu ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, ale właściwość ma wartość NULL. W takim przypadku właściwość jest usuwana z komunikatu.

Dane komunikatu do przekazania są pobierane z parametru buforu MQPUT/MQPUT1 .

RAPORT MQACTP_REPORT

Raport jest generowany w wyniku wcześniej pobranego komunikatu. Oryginalny uchwyt komunikatu określa komunikat powodujący wygenerowanie raportu.

Nowy uchwyt komunikatu określa wszelkie modyfikacje właściwości (w tym dowolnej w deskryptorze komunikatu) w oryginalnym uchwycie komunikatu.

Deskryptor komunikatu składa się z następujących elementów:

- Jeśli parametr MsgDesc jest podany w wywołaniu MQPUT lub MQPUT1 , a parametr MQPMO_MD_FOR_OUTPUT_ONLY nie znajduje się w MQPMO.Optionsjest używana jako niezmodyfikowany deskryptor komunikatu.
- Jeśli parametr MsgDesc nie został podany lub jeśli parametr MQPMO_MD_FOR_OUTPUT_ONLY znajduje się w produkcie MQPMO.Options , a następnie pola początkowego deskryptora komunikatu są wybierane w następujący sposób:

Tabela 510. Transformacja uchwytu komunikatu raportu

Pole w strukturze MQMD	**Użyta wartość**
Raport	Jeśli MQRO_PASS_DISCARD_AND_TERMIN i Ustawiono MQRO_DISCARD_MSG: MQRO_DISCARD_MSG w przeciwnym razie MQRO_BRAK
MsgType	MQMT_RAPORT
Utrata ważności	Jeśli MQRO_PASS_DISCARD_AND_WAŻNOŚCI jest ustawione: Skopiowane z komunikatu wejściowego w przeciwnym razie MQEI_UNLIMITED,
MsgId	Jeśli ustawiono MQPMO_NEW_MSG_ID: Generowany jest nowy identyfikator komunikatu w przeciwnym razie, jeśli ustawiono MQRO_PASS_MSG_ID: Skopiowane z komunikatu wejściowego w przeciwnym razie MQMI_BRAK

Tabela 510. Transformacja uchwytu komunikatu raportu (kontynuacja)	
Pole w strukturze MQMD	**Użyta wartość**
CorrelId	Jeśli ustawiono wartość MQPMO_NEW_CORREL_ID: Zostanie wygenerowany nowy identyfikator korelacji w przeciwnym razie, jeśli ustawiono MQRO_COPY_MSG_ID_TO_CORREL_ID: Skopiowany z pola MsgId w Komunikat wejściowy w przeciwnym razie, jeśli ustawiono MQRO_PASS_CORREL_ID: Skopiowane z pola CorrelId Komunikat wejściowy w przeciwnym razie MQCI_BRAK
BackoutCount	0
ReplyToQ	Puste
ReplyToQMgr	Puste
OriginalLength	Ustaw na *BufferLength*

- Deskryptor komunikatu jest następnie modyfikowany przez nowy uchwyt komunikatu-wszystkie pola deskryptora komunikatu jawnie ustawione jako właściwości w nowym uchwycie komunikatu mają pierwszeństwo przed polami deskryptora komunikatu zgodnie z wcześniejszym opisem.

Właściwości komunikatu są tworzone w następujący sposób:

- Wszystkie właściwości z oryginalnego uchwytu komunikatu, które mają wartość MQCOPY_REPORT w MQPD.CopyOptions
- Wszystkie właściwości z nowego uchwytu komunikatu. Dla każdej właściwości w nowym uchwycie komunikatu, która ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, wartość jest pobierana z nowego uchwytu komunikatu. Jedynym wyjątkiem od tej reguły jest przypadek specjalny, gdy właściwość w nowym uchwycie komunikatu ma taką samą nazwę jak właściwość w oryginalnym uchwycie komunikatu, ale właściwość ma wartość NULL. W takim przypadku właściwość jest usuwana z komunikatu.

Pole Feedback w wynikowej strukturze MQMD reprezentuje raport, który ma zostać wygenerowany. Wartość parametru Feedback równa MQFB_NONE powoduje niepowodzenie wywołania MQPUT lub MQPUT1 z kodem przyczyny MQRC_FEEDBACK_ERROR.

Aby wybrać dane użytkownika dla komunikatu raportu, IBM MQ sprawdza pola Raport i Opinia w wynikowej strukturze MQMD oraz parametry Buffer i BufferLength wywołania MQPUT lub MQPUT1 .

- Jeśli Opinia to MQFB_COA, MQFB_COD lub MQFB_EXPIRATION, wartość raportu jest sprawdzana.
- Jeśli spełniony jest którykolwiek z poniższych warunków, używane są pełne dane komunikatu z buforu o długości BufferLength .
  - Informacja zwrotna to MQFB_EXPIRATION, a raport zawiera dane MQRO_EXPIRATION_WITH_FULL_DATA
  - Informacja zwrotna to MQFB_COD, a raport zawiera MQRO_COD_WITH_FULL_DATA
  - Informacja zwrotna to MQFB_COA, a raport zawiera MQRO_COA_WITH_FULL_DATA
- Jeśli spełniony jest którykolwiek z poniższych warunków, używane jest pierwsze 100 bajtów komunikatu (lub wartość BufferLength , jeśli jest mniejsza niż 100) z buforu.
  - Informacja zwrotna to MQFB_EXPIRATION, a raport zawiera dane MQRO_EXPIRATION_WITH_DATA
  - Informacja zwrotna to MQFB_COD, a raport zawiera MQRO_COD_WITH_DATA

  - Informacja zwrotna to MQFB_COA, a raport zawiera MQRO_COA_WITH_DATA
- Jeśli baza danych Feedback ma wartość MQFB_EXPIRATION, MQFB_COD lub MQFB_COA, a raport nie zawiera opcji * _WITH_FULL_DATA lub * _WITH_DATA dotyczących tej wartości Feeback, komunikat nie będzie zawierał żadnych danych użytkownika.
- Jeśli baza danych Feedback przyjmuje inną wartość niż wymienione powyżej, to w normalnej sytuacji używane są bufory i BufferLength .

W poniższej tabeli przedstawiono również wyprowadzenie danych użytkownika opisanych na poprzedniej liście:

Tabela 511. Źródło danych użytkownika

	MQFB_COA	**MQFB_COD**	**MQFB_UTR_WAŻN.**
MQRO_EXPIRATION _WITH_FULL_DATA	Brak	Brak	Bufor (długość buforu)
MQRO_COD_WITH _FULL_DATA	Brak	Bufor (długość buforu)	Brak
MQRO_COA_WITH _DANE	Bufor (długość buforu)	Brak	Brak
MQRO_EXPIRATION _WITH_DATA	Brak	Brak	Bufor (pierwsze 100 bajtów)
MQRO_KOD_WITH_DA TA	Brak	Bufor (pierwsze 100 bajtów)	Brak
MQRO_KOA_WITH_DA TA	Bufor (pierwsze 100 bajtów)	Brak	Brak

PubLevel (MQLONG) dla MQPMO

Wartością początkową tego pola jest 9. Poziom subskrypcji, której dotyczy ta publikacja. Tę publikację otrzymują tylko te subskrypcje, których najwyższy SubLevel jest mniejszy lub równy tej wartości. Wartość ta musi mieścić się w zakresie od 0 do 9; zero jest najniższym poziomem. Jeśli jednak publikacja została zachowana, nie jest już dostępna dla subskrybentów na wyższych poziomach, ponieważ jest ponownie publikowana na poziomie PubLevel 1.

Więcej informacji na ten temat zawiera sekcja Przechwytywanie publikacji.

MQPMR-rekord komunikatu umieszczania

Struktura MQPMR umożliwia określenie różnych właściwości komunikatu dla pojedynczego miejsca docelowego podczas umieszczania komunikatu na liście dystrybucyjnej. MQPMR to struktura wejścia/wyjścia dla wywołań MQPUT i MQPUT1 .

Dostępność

Struktura MQPMR jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Zestaw znaków i kodowanie

Dane w rekordzie MQPMR muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Użycie

Udostępniając tablicę tych struktur w wywołaniu MQPUT lub MQPUT1 , można określić różne wartości dla każdej kolejki docelowej na liście dystrybucyjnej. Niektóre pola są tylko polami wejściowymi, inne są polami wejściowymi i wyjściowymi.

Uwaga: Ta struktura jest nietypowa, ponieważ nie ma stałego układu. Pola w tej strukturze są opcjonalne, a obecność lub nieobecność każdego pola jest wskazywana przez flagi w polu *PutMsgRecFields* w MQPMO. Pola, które są obecne w produkcie ***, muszą występować w następującej kolejności*** :

- *MsgId*
- *CorrelId*
- *GroupId*
- *Feedback*
- *AccountingToken*

Nieobecne pola nie zajmują miejsca w rekordzie.

Ponieważ raport MQPMR nie ma stałego układu, w plikach nagłówka, COPY i INCLUDE dla obsługiwanych języków programowania nie jest dostępna jego definicja. Programista aplikacji musi utworzyć deklarację zawierającą pola, które są wymagane przez aplikację, i ustawić flagi w pliku *PutMsgRecFields* , aby wskazać pola, które są obecne.

Pola

Dla tej struktury nie zdefiniowano żadnych wartości początkowych, ponieważ w plikach nagłówka, COPYi INCLUDE dla obsługiwanych języków programowania nie ma deklaracji struktury. Przykładowe deklaracje przedstawiają sposób deklarowania struktury, jeśli wszystkie pola są wymagane.

Tabela 512. Pola w MQPMR

Nazwa pola	**Opis pola**
MsgId	Identyfikator komunikatu
CorrelId	Identyfikator korelacji
GroupId	Identyfikator grupy
Opinie	Informacja zwrotna lub kod przyczyny
AccountingToken	Token rozliczania

Deklaracje językowe

Deklaracja C dla MQPMR

```
typedef struct tagMQPMR MQPMR;
struct tagMQPMR {
  MQBYTE24  MsgId;            /* Message identifier */
  MQBYTE24  CorrelId;         /* Correlation identifier */
  MQBYTE24  GroupId;          /* Group identifier */
  MQLONG    Feedback;         /* Feedback or reason code */
  MQBYTE32  AccountingToken;  /* Accounting token */
};
```

Deklaracja COBOL dla MQPMR

```
**   MQPMR structure
  10 MQPMR.
**    Message identifier
   15 MQPMR-MSGID           PIC X(24).
**    Correlation identifier
   15 MQPMR-CORRELID        PIC X(24).
**    Group identifier
   15 MQPMR-GROUPID         PIC X(24).
**    Feedback or reason code
   15 MQPMR-FEEDBACK        PIC S9(9) BINARY.
**    Accounting token
   15 MQPMR-ACCOUNTINGTOKEN PIC X(32).
```

Deklaracja PL/I dla MQPMR

```
dcl
 1 MQPMR based,
   3 MsgId           char(24),      /* Message identifier */
   3 CorrelId        char(24),      /* Correlation identifier */
   3 GroupId         char(24),      /* Group identifier */
   3 Feedback        fixed bin(31), /* Feedback or reason code */
   3 AccountingToken char(32);      /* Accounting token */
```

Deklaracja Visual Basic dla MQPMR

```
Type MQPMR
  MsgId           As MQBYTE24 'Message identifier'
  CorrelId        As MQBYTE24 'Correlation identifier'
  GroupId         As MQBYTE24 'Group identifier'
  Feedback        As Long     'Feedback or reason code'
  AccountingToken As MQBYTE32 'Accounting token'
End Type
```

MsgId (MQBYTE24) w rekordzie MQPMR

Jest to identyfikator komunikatu, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *MsgId* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie występuje w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowych, wartość w MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *MsgId* . Jeśli ta wartość to MQMI_NONE, dla *każdego* z tych miejsc docelowych generowany jest nowy identyfikator komunikatu (czyli żadne dwa z tych miejsc docelowych nie mają tego samego identyfikatora komunikatu).

Jeśli podano parametr MQPMO_NEW_MSG_ID, nowe identyfikatory komunikatów są generowane dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania MQPMO_NEW_CORREL_ID (patrz pole *CorrelId*).

Jest to pole wejściowe/wyjściowe.

CorrelId (MQBYTE24) dla MQPMR

Jest to identyfikator korelacji, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *CorrelId* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie występuje w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowych, wartość w MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *CorrelId* .

Jeśli określono parametr MQPMO_NEW_CORREL_ID, *pojedynczy* nowy identyfikator korelacji jest generowany i używany dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania MQPMO_NEW_MSG_ID (patrz pole *MsgId*).

Jest to pole wejściowe/wyjściowe.

GroupId (MQBYTE24) dla MQPMR

GroupId to identyfikator grupy, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *GroupId* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie występuje w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowych, wartość w MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *GroupId* . Wartość jest przetwarzana zgodnie z opisem w sekcji Kolejność fizyczna w kolejce, ale z następującymi różnicami:

- GroupId jest tworzony na podstawie nazwy QMName i znacznika czasu. Dlatego, aby zachować unikalny identyfikator grupy (GroupId), należy zachować również unikalne nazwy menedżerów kolejek. Nie należy również ustawiać zegarów z powrotem na komputerze menedżera kolejek.
- W przypadku użycia nowego identyfikatora grupy menedżer kolejek generuje inny identyfikator grupy dla każdego miejsca docelowego (tzn. żadne dwa miejsca docelowe nie mają tego samego identyfikatora grupy).
- W przypadku użycia wartości w polu wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_GROUP_ID_ERROR.

Jest to pole wejściowe/wyjściowe.

Informacje zwrotne (MQLONG) dla MQPMR

Jest to kod informacji zwrotnej, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *Feedback* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

Jest to pole wejściowe.

AccountingToken (MQBYTE32) dla MQPMR

Jest to znacznik rozliczania, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *AccountingToken* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce. Więcej informacji na temat zawartości tego pola zawiera opis *AccountingToken* w sekcji “MQMD-deskryptor komunikatu” na stronie 432 .

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

Jest to pole wejściowe.

MQRFH-reguły i nagłówek formatowania

Struktura MQRFH definiuje układ reguł i nagłówka formatowania. Ten nagłówek służy do wysyłania danych łańcuchowych w postaci par nazwa-wartość.

Dostępność

Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Nazwa formatu

ZMQFMT_RF_HEADER

Zestaw znaków i kodowanie

Pola w strukturze MQRFH (w tym *NameValueString*) znajdują się w zestawie znaków i kodowaniu określonym przez pola *CodedCharSetId* i *Encoding* w strukturze nagłówka poprzedzającej strukturę MQRFH lub przez te pola w strukturze MQMD, jeśli MQRFH znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 513. Pola w MQRFH dla MQRFH

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQRFH_ID_struktury	'RFH¬'
Wersja (numer wersji struktury)	MQRFH_VERSION_1	1
StrucLength (długość w bajtach struktury MQRFH)	MQRFH_STRUC_LENGTH_FIXED	32
Kodowanie (kodowanie liczbowe danych następujących po *NameValueString*)	RODZIMA MQENC	Zależy od środowiska
CodedCharSetId (określa identyfikator zestawu znaków danych, które następują po *NameValueString*)	MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0
Format (nazwa formatu danych następujących po *NameValueString*)	MQFMT_BRAK	Puste
Flagi (flags)	MQRFH_BRAK	0
NameValueString (łańcuch znaków o zmiennej długości zawierający pary nazwa-wartość)	brak	brak

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQRFH_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQRFH MyRFH = {MQRFH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQRFH

```
typedef struct tagMQRFH MQRFH;
struct tagMQRFH {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
```

```
  MQLONG   StrucLength;     /* Total length of MQRFH including
                               NameValueString */
  MQLONG   Encoding;        /* Numeric encoding of data that follows
                               NameValueString */
  MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows NameValueString */
  MQCHAR8  Format;          /* Format name of data that follows
                               NameValueString */
  MQLONG   Flags;           /* Flags */
};
```

Deklaracja języka COBOL dla MQRFH

```
**   MQRFH structure
  10 MQRFH.
**    Structure identifier
   15 MQRFH-STRUCID        PIC X(4).
**    Structure version number
   15 MQRFH-VERSION        PIC S9(9) BINARY.
**    Total length of MQRFH including NAMEVALUESTRING
   15 MQRFH-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows NAMEVALUESTRING
   15 MQRFH-ENCODING       PIC S9(9) BINARY.
**    Character set identifier of data that follows NAMEVALUESTRING
   15 MQRFH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows NAMEVALUESTRING
   15 MQRFH-FORMAT         PIC X(8).
**    Flags
   15 MQRFH-FLAGS          PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQRFH

```
dcl
 1 MQRFH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Total length of MQRFH including
                                     NameValueString */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows NameValueString */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows NameValueString */
  3 Format         char(8),       /* Format name of data that follows
                                     NameValueString */
  3 Flags          fixed bin(31); /* Flags */
```

Deklaracja High Level Assembler dla MQRFH

```
MQRFH                 DSECT
MQRFH_STRUCID         DS   CL4  Structure identifier
MQRFH_VERSION         DS   F    Structure version number
MQRFH_STRUCLENGTH     DS   F    Total length of MQRFH including
*                               NAMEVALUESTRING
MQRFH_ENCODING        DS   F    Numeric encoding of data that follows
*                               NAMEVALUESTRING
MQRFH_CODEDCHARSETID  DS   F    Character set identifier of data that
*                               follows NAMEVALUESTRING
MQRFH_FORMAT          DS   CL8  Format name of data that follows
*                               NAMEVALUESTRING
MQRFH_FLAGS           DS   F    Flags
*
MQRFH_LENGTH          EQU  *-MQRFH
                      ORG  MQRFH
MQRFH_AREA            DS   CL(MQRFH_LENGTH)
```

Deklaracja Visual Basic dla MQRFH

```
Type MQRFH
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Total length of MQRFH including'
                             'NameValueString'
  Encoding       As Long     'Numeric encoding of data that follows'
```

```
                              'NameValueString'
  CodedCharSetId As Long      'Character set identifier of data that'
                              'follows NameValueString'
  Format         As String*8  'Format name of data that follows'
                              'NameValueString'
  Flags          As Long      'Flags'
End Type
```

StrucId (MQCHAR4) dla MQRFH

Jest to identyfikator struktury reguł i struktury nagłówka formatowania. Jest to zawsze pole wejściowe. Jego wartością jest MQRFH_STRUC_ID.

Wartość musi być następująca:

MQRFH_ID_struktury
Identyfikator struktury nagłówka reguł i formatowania.

Dla języka programowania C zdefiniowana jest również stała MQRFH_STRUC_ID_ARRAY. Ma taką samą wartość jak MQRFH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQRFH

Jest to numer wersji struktury; wartość musi być następująca:

MQRFH_VERSION_1
Version-1 -reguły i struktura nagłówka formatowania.

Wartością początkową tego pola jest MQRFH_VERSION_1.

StrucLength (MQLONG) dla MQRFH

Jest to długość (w bajtach) struktury MQRFH, łącznie z polem *NameValueString* na końcu struktury. Długość nie obejmuje żadnych danych użytkownika, które następują po polu *NameValueString* .

Aby uniknąć problemów z przekształcaniem danych użytkownika w niektórych środowiskach, *StrucLength* musi być wielokrotnością czterech.

Następująca stała określa długość *stałej* części struktury, czyli długość bez pola *NameValueString* :

MQRFH_STRUC_LENGTH_FIXED
Długość stałej części struktury MQRFH.

Wartością początkową tego pola jest MQRFH_STRUC_LENGTH_FIXED.

Kodowanie (MQLONG) dla MQRFH

Określa kodowanie liczbowe danych następujących po *NameValueString* ; nie ma zastosowania do danych liczbowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest MQENC_NATIVE.

CodedCharSetId (MQLONG) dla MQRFH

Określa identyfikator zestawu znaków danych następujących po *NameValueString* ; nie ma zastosowania do danych znakowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Pola MQCCSI_INHERIT nie można używać, jeśli wartością pola *PutApplType* w deskryptorze MQMD jest MQAT_BROKER.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQRFH

Określa nazwę formatu danych następujących po *NameValueString*.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQRFH

Można określić następujące elementy:

MQRFH_BRAK
Brak flag.

Wartością początkową tego pola jest MQRFH_NONE.

NameValueŁańcuch (MQCHARn) dla MQRFH

Jest to łańcuch znaków o zmiennej długości, zawierający pary nazwa-wartość w postaci:

```
name1 value1 name2 value2 name3 value3 ...
```

Każda nazwa lub wartość musi być oddzielona od sąsiedniej nazwy lub wartości jednym lub większą liczbą znaków odstępu. Te znaki odstępu nie są istotne. Nazwa lub wartość może zawierać znaczące odstępy, poprzedzone i poprzedzone nazwą lub wartością podwójnymi cudzysłowami; wszystkie znaki między otwierającym i zamykającym znakiem cudzysłowu są traktowane jako znaczące. W poniższym przykładzie nazwa to FAMOUS_WORDS, a wartość to `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Nazwa lub wartość może zawierać dowolne znaki inne niż znak o kodzie zero (który działa jako separator w przypadku *NameValueString*). Jednak w celu ułatwienia współdziałania aplikacja może ograniczyć nazwy do następujących znaków:

- Pierwszy znak: wielkie lub małe litery (od A do Z lub od a do z) lub podkreślenie.
- Kolejne znaki: wielka lub mała litera, cyfra dziesiętna (od 0 do 9), podkreślenie, łącznik lub kropka.

Jeśli nazwa lub wartość zawiera jeden lub więcej podwójnych cudzysłowów, nazwa lub wartość musi być ujęta w podwójne cudzysłowy, a każdy podwójny cudzysłów w łańcuchu musi być podwojony:

```
Famous_Words "The program displayed ""Hello World"""
```

W nazwach i wartościach rozróżniana jest wielkość liter, tzn. małe litery nie są traktowane jako wielkie litery. Na przykład FAMOUS_WORDS i `Famous_Words` to dwie różne nazwy.

Długość w bajtach *NameValueString* jest równa *StrucLength* minus MQRFH_STRUC_LENGTH_FIXED. Aby uniknąć problemów z przekształcaniem danych użytkowników w niektórych środowiskach, należy ustawić tę długość jako wielokrotność liczby czterech. Należy dopełnić łańcuch *NameValueString* spacjami do tej długości lub zakończyć go wcześniej, umieszczając znak o kodzie zero po ostatnim znaczącym znaku w łańcuchu. Znak o kodzie zero i następujące po nim bajty (do określonej długości *NameValueString*) są ignorowane.

Uwaga: Ponieważ długość tego pola nie jest stała, pole jest pomijane w deklaracjach struktury, które są udostępniane dla obsługiwanych języków programowania.

MQRFH2 -reguły i nagłówek formatowania 2

Nagłówek MQRFH2 jest oparty na nagłówku MQRFH , ale umożliwia przesyłanie łańcuchów Unicode bez translacji i może przenosić liczbowe typy danych.Struktura MQRFH2 definiuje format reguł i nagłówka formatowania version-2 . Ten nagłówek służy do wysyłania danych, które zostały zakodowane przy użyciu składni podobnej do składni XML. Komunikat może zawierać dwie lub więcej struktur MQRFH2 w serii, z danymi użytkownika opcjonalnie po ostatniej strukturze MQRFH2 w serii.

Dostępność

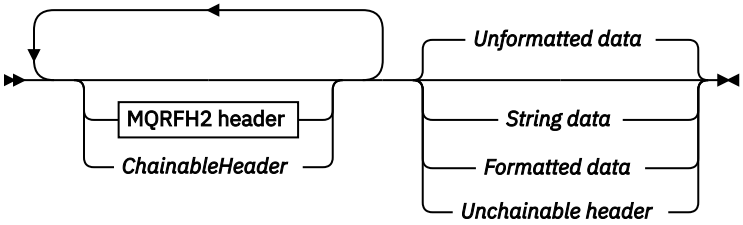
Wszystkie systemy IBM MQ oraz IBM MQ MQI clients połączone z tymi systemami.

Nazwa formatu

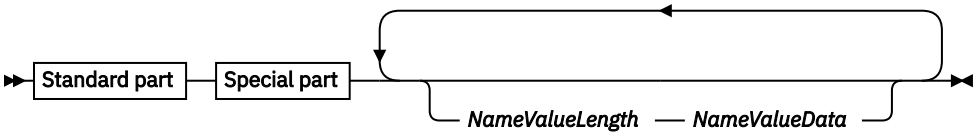
MQFMT_RF_HEADER_2

Syntax

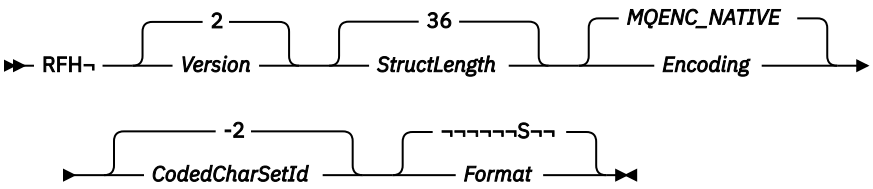
IBM MQ Message



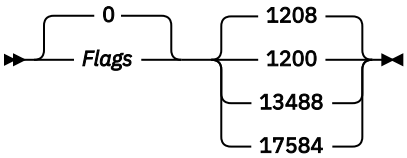
MQRFH2 header



Standard part



Special part



Zestaw znaków i kodowanie

Specjalne reguły mają zastosowanie do zestawu znaków i kodowania używanego w strukturze MQRFH2 :

- Pola inne niż *NameValueData* mają zestaw znaków i kodowanie określone przez pola *CodedCharSetId* i *Encoding* w strukturze nagłówka, która poprzedza MQRFH2, lub przez te pola w strukturze MQMD , jeśli zmienna MQRFH2 znajduje się na początku danych komunikatu aplikacji.

  Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Jeśli w wywołaniu MQGET określono wartość MQGMO_CONVERT , menedżer kolejek przekształca pola MQRFH2 inne niż *NameValueData*w żądany zestaw znaków i kodowanie.

- *NameValueData* znajduje się w zestawie znaków podanym w polu *NameValueCCSID* . Tylko wymienione zestawy znaków Unicode są poprawne dla *NameValueCCSID* ; Szczegółowe informacje można znaleźć w opisie komendy *NameValueCCSID* .

  Niektóre zestawy znaków mają reprezentację zależną od kodowania. Jeśli *NameValueCCSID* jest jednym z tych zestawów znaków, kodowanie *NameValueData* musi być takie samo jak kodowanie innych pól w MQRFH2.

  Jeśli w wywołaniu komendy MQGET określono opcję MQGMO_CONVERT , menedżer kolejek przekształca wartość *NameValueData* w żądane kodowanie, ale nie zmienia swojego zestawu znaków.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 514. Pola w nagłówku MQRFH2 dla MQRFH2

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId (identyfikator struktury)	MQRFH_ID_struktury	'RFH¬'
Wersja (numer wersji struktury)	MQRFH_VERSION_2	2
StrucLength (długość w bajtach struktury MQRFH2)	MQRFH_STRUC_LENGTH_FIXED_2	36
Kodowanie (kodowanie liczbowe danych następujących po ostatnim polu *NameValueData*)	RODZIMA MQENC	Zależy od środowiska
CodedCharSetId (identyfikator zestawu znaków danych następujących po ostatnim polu *NameValueData*)	MQCCSI_INHERIT	-2
Format (nazwa formatu danych po ostatnim polu *NameValueData*)	MQFMT_BRAK	Puste
Flagi (flags)	MQRFH_BRAK	0
NameValueCCSID (identyfikator kodowanego zestawu znaków dla danych w polu *NameValueData*)	Brak	1208
NameValueDługość (długość w bajtach danych w polu *NameValueData*)	Brak	None

Tabela 514. Pola w nagłówku MQRFH2 dla MQRFH2 (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
NameValueDane (pary nazwa-wartość właściwości komunikatu)	Brak	Brak

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQRFH2_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQRFH2 MyRFH2 = {MQRFH2_DEFAULT};
```

Deklaracje językowe

Deklaracja języka C dla MQRFH2

```
typedef struct tagMQRFH2 MQRFH2;
struct tagMQRFH2 {
  MQCHAR4  StrucId;         /* Structure identifier */
  MQLONG   Version;         /* Structure version number */
  MQLONG   StrucLength;     /* Total length of MQRFH2 including all
                               NameValueLength and NameValueData
                               fields */
  MQLONG   Encoding;        /* Numeric encoding of data that follows
                               last NameValueData field */
  MQLONG   CodedCharSetId;  /* Character set identifier of data that
                               follows last NameValueData field */
  MQCHAR8  Format;          /* Format name of data that follows last
                               NameValueData field */
  MQLONG   Flags;           /* Flags */
  MQLONG   NameValueCCSID;  /* Character set identifier of
                               NameValueData */
};
```

Deklaracja języka COBOL dla MQRFH2

```
**   MQRFH2 structure
  10 MQRFH2.
**    Structure identifier
   15 MQRFH2-STRUCID        PIC X(4).
**    Structure version number
   15 MQRFH2-VERSION        PIC S9(9) BINARY.
**    Total length of MQRFH2 including all NAMEVALUELENGTH and
**    NAMEVALUEDATA fields
   15 MQRFH2-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows last NAMEVALUEDATA field
   15 MQRFH2-ENCODING       PIC S9(9) BINARY.
**    Character set identifier of data that follows last NAMEVALUEDATA
**    field
   15 MQRFH2-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows last NAMEVALUEDATA field
   15 MQRFH2-FORMAT         PIC X(8).
**    Flags
   15 MQRFH2-FLAGS          PIC S9(9) BINARY.
**    Character set identifier of NAMEVALUEDATA
   15 MQRFH2-NAMEVALUECCSID PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQRFH2

```
dcl
 1 MQRFH2 based,
  3 StrucId        char(4),       /* Structure identifier */
```

```
  3 Version        fixed bin(31), /* Structure version number */
  3 StrucLength    fixed bin(31), /* Total length of MQRFH2 including
                                     all NameValueLength and
                                     NameValueData fields */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows last NameValueData field */
  3 CodedCharSetId fixed bin(31), /* Character set identifier of data
                                     that follows last NameValueData
                                     field */
  3 Format         char(8),       /* Format name of data that follows
                                     last NameValueData field */
  3 Flags          fixed bin(31), /* Flags */
  3 NameValueCCSID fixed bin(31); /* Character set identifier of
                                     NameValueData */
```

Deklaracja High Level Assembler dla MQRFH2

```
MQRFH                 DSECT
MQRFH_STRUCID         DS   CL4  Structure identifier
MQRFH_VERSION         DS   F    Structure version number
MQRFH_STRUCLENGTH     DS   F    Total length of MQRFH2 including all
*                               NAMEVALUELENGTH and NAMEVALUEDATA fields
MQRFH_ENCODING        DS   F    Numeric encoding of data that follows
*                               last NAMEVALUEDATA field
MQRFH_CODEDCHARSETID  DS   F    Character set identifier of data that
*                               follows last NAMEVALUEDATA field
MQRFH_FORMAT          DS   CL8  Format name of data that follows last
*                               NAMEVALUEDATA field
MQRFH_FLAGS           DS   F    Flags
MQRFH_NAMEVALUECCSID  DS   F    Character set identifier of
*                               NAMEVALUEDATA
*
MQRFH_LENGTH          EQU  *-MQRFH
                      ORG  MQRFH
MQRFH_AREA            DS   CL(MQRFH_LENGTH)
```

Deklaracja Visual Basic dla MQRFH2

```
Type MQRFH2
  StrucId        As String*4 'Structure identifier'
  Version        As Long     'Structure version number'
  StrucLength    As Long     'Total length of MQRFH2 including all'
                             'NameValueLength and NameValueData fields'
  Encoding       As Long     'Numeric encoding of data that follows'
                             'last NameValueData field'
  CodedCharSetId As Long     'Character set identifier of data that'
                             'follows last NameValueData field'
  Format         As String*8 'Format name of data that follows last'
                             'NameValueData field'
  Flags          As Long     'Flags'
  NameValueCCSID As Long     'Character set identifier of NameValueData'
End Type
```

StrucId (MQCHAR4) dla MQRFH2

Jest to identyfikator struktury dwóch struktur nagłówka reguł i formatowania. Jest to zawsze pole wejściowe. Jego wartością jest MQRFH2_STRUC_ID.

Wartość musi być następująca:

MQRFH2_STRUC_ID
Identyfikator struktury 2 nagłówka reguł i formatowania.

W języku programowania C zdefiniowana jest również stała MQRFH2_STRUC_ID_ARRAY . Ma taką samą wartość jak MQRFH2_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQRFH2

Jest to numer wersji struktury; wartość musi być następująca:

MQRFH_VERSION_2
Reguły i struktura nagłówka formatowania Version-2 .

Wartością początkową tego pola jest MQRFH_VERSION_2.

StrucLength (MQLONG) dla MQRFH2

Jest to długość w bajtach struktury MQRFH2 z uwzględnieniem pól *NameValueLength* i *NameValueData* na końcu struktury. Poprawne jest, że na końcu struktury znajduje się wiele par pól *NameValueLength* i *NameValueData* w sekwencji:

```
length1, data1, length2, data2, ...
```

StrucLength nie zawiera żadnych danych użytkownika, które mogą występować po ostatnim polu *NameValueData* na końcu struktury.

Aby uniknąć problemów z przekształcaniem danych użytkownika w niektórych środowiskach, *StrucLength* musi być wielokrotnością czterech.

Następująca stała określa długość *stałej* części struktury, czyli długość z wyłączeniem pól *NameValueLength* i *NameValueData* :

MQRFH_STRUC_LENGTH_FIXED_2
Długość stałej części struktury MQRFH2 .

Wartością początkową tego pola jest MQRFH_STRUC_LENGTH_FIXED_2.

Kodowanie (MQLONG) dla MQRFH2

Określa kodowanie liczbowe danych następujących po ostatnim polu *NameValueData* . Nie ma ono zastosowania do danych liczbowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest MQENC_NATIVE.

CodedCharSetId (MQLONG) dla MQRFH2

Służy do określania identyfikatora zestawu znaków danych następującego po ostatnim polu *NameValueData* . Nie ma on zastosowania do danych znakowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Pola MQCCSI_INHERIT nie można używać, jeśli wartością pola *PutApplType* w deskryptorze MQMD jest MQAT_BROKER.

Wartością początkową tego pola jest MQCCSI_INHERIT.

Format (MQCHAR8) dla MQRFH2

Określa nazwę formatu danych następujących po ostatnim polu *NameValueData* .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

Opcje (MQLONG) dla MQRFH2

Wartością początkową tego pola jest MQRFH_NONE. MQRFH_NONE należy podać.

MQRFH_NONE
Brak flag.

MQRFH_INTERNAL
Nagłówek MQRFH2 zawiera wewnętrznie ustawione właściwości.

MQRFH_INTERNAL jest przeznaczony do użycia przez menedżer kolejek.

Pierwsze 16 bitów, MQRFH_FLAGS_RESTRICTED_MASK, są zarezerwowane dla flag zestawów menedżerów kolejek. Flagi, które użytkownik może ustawić, są zdefiniowane w 16 dolnych bitach.

NameValueCCSID (MQLONG) dla MQRFH2

Określa identyfikator kodowanego zestawu znaków dla danych w polu *NameValueData* . Różni się to od zestawu znaków innych łańcuchów w strukturze MQRFH2 i może różnić się od zestawu znaków danych (jeśli istnieje), który występuje po ostatnim polu *NameValueData* na końcu struktury.

NameValueCCSID musi mieć jedną z następujących wartości:

CCSID	**Znaczenie**
1200	UTF-16, najnowsza obsługiwana wersja Unicode
13488	UTF-16, podzbiór Unicode w wersji 2.0
17584	Podzbiór UTF-16, wersja Unicode 3.0 (zawiera symbol euro)
1208	UTF-8, najnowsza obsługiwana wersja Unicode

W przypadku zestawów znaków UTF-16 kodowanie (kolejność bajtów) w pliku *NameValueData* musi być takie samo jak kodowanie innych pól w strukturze MQRFH2 .

Znaki wykraczające poza kod Unicode Basic Multilingual Plane (te powyżej U + FFFF), reprezentowane w UTF-16 przez odpowiedniki punktów kodowych (X'D800'do X'DFFF') lub cztery bajty w UTF-8, nie są obsługiwane.

Uwaga: Jeśli produkt *NameValueCCSID* nie ma jednej z wymienionych powyżej wartości, a struktura MQRFH2 wymaga konwersji w wywołaniu MQGET, wywołanie zostanie zakończone z kodem przyczyny MQRC_SOURCE_CCSID_ERROR, a komunikat zostanie zwrócony bez konwersji.

Wartością początkową tego pola jest 1208.

NameValueDługość (MQLONG) dla MQRFH2

Długość odpowiedniego pola NameValueData

Określa długość (w bajtach) danych w polu *NameValueData* . *NameValueLength* musi być wielokrotnością czterech.

Uwaga: Pola *NameValueLength* i *NameValueData* są opcjonalne, ale jeśli występują, muszą występować jako para i być sąsiadujące. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

```
length1 data1 length2 data2 length3 data3
```

Ponieważ te pola są opcjonalne, są pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.

NameValueDane (MQCHARn) dla MQRFH2

NameValueData to pole o zmiennej długości, które zawiera folder zawierający pary nazwa-wartość właściwości komunikatu. Folder jest łańcuchem znaków o zmiennej długości, zawierającym dane zakodowane przy użyciu składni podobnej do składni XML. Długość łańcucha znaków w bajtach jest określona przez pole *NameValueLength* , które poprzedza pole *NameValueData* . Długość musi być wielokrotnością liczby czterech.

Pola *NameValueLength* i *NameValueData* są opcjonalne, ale jeśli występują, muszą występować jako para i być sąsiadujące. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

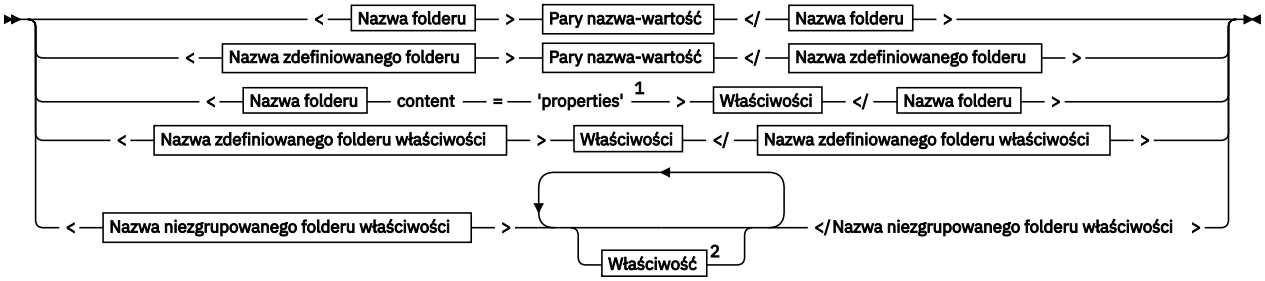
```
length1 data1 length2 data2 length3 data3
```

NameValueData nie jest konwertowany na zestaw znaków określony w wywołaniu MQGET . Nawet jeśli komunikat jest pobierany z obowiązującą opcją MQGMO_CONVERT *NameValueData* , pozostaje w oryginalnym zestawie znaków. Jednak kodowanie *NameValueData* jest konwertowane na kodowanie określone w wywołaniu MQGET .

Uwagi:

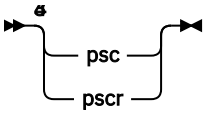
- Ponieważ te pola są opcjonalne, są pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.
- W diagramie składni używane są terminy "zdefiniowane" i "zastrzeżone" . "Zdefiniowane" oznacza, że nazwa jest używana przez IBM MQ. "Zarezerwowana" oznacza, że nazwa jest zarezerwowana dla produktu IBM MQw przyszłości.

NameValueData Składnia

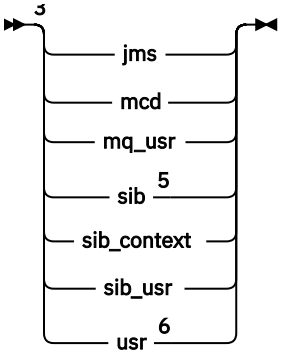


Nazwa folderu

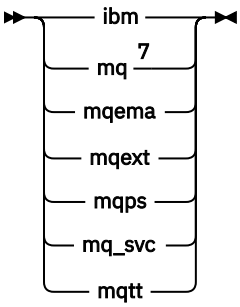
Nazwa zdefiniowanego folderu



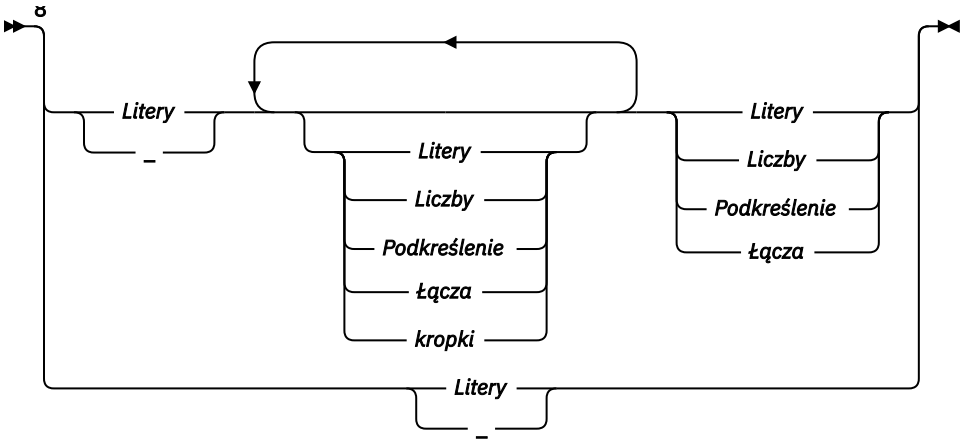
Nazwa zdefiniowanego folderu właściwości



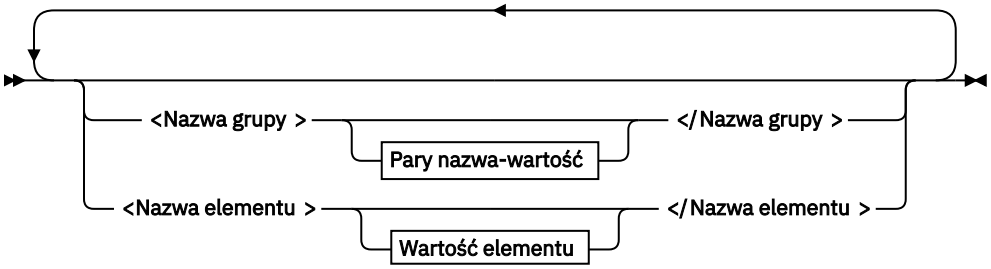
Nazwa niezgrupowanego folderu właściwości



Nazwa



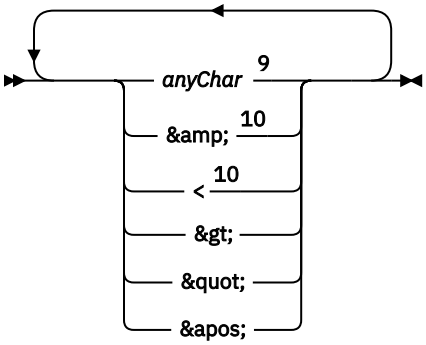
Pary nazwa-wartość



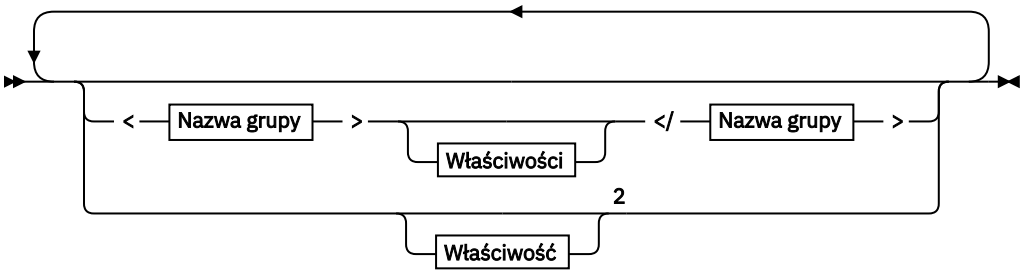
Nazwa grupy

Nazwa elementu

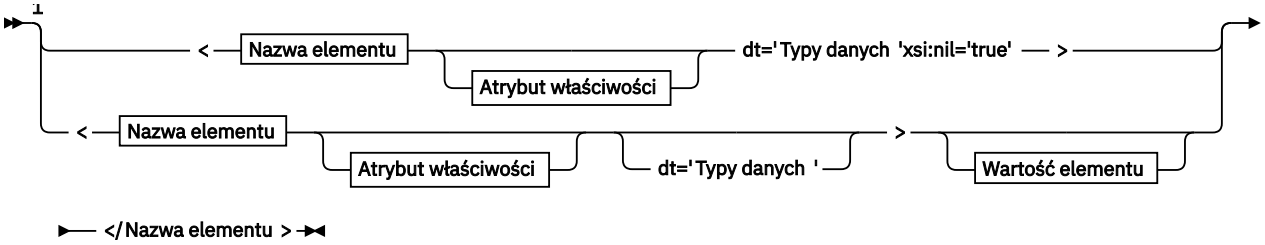
Wartość elementu



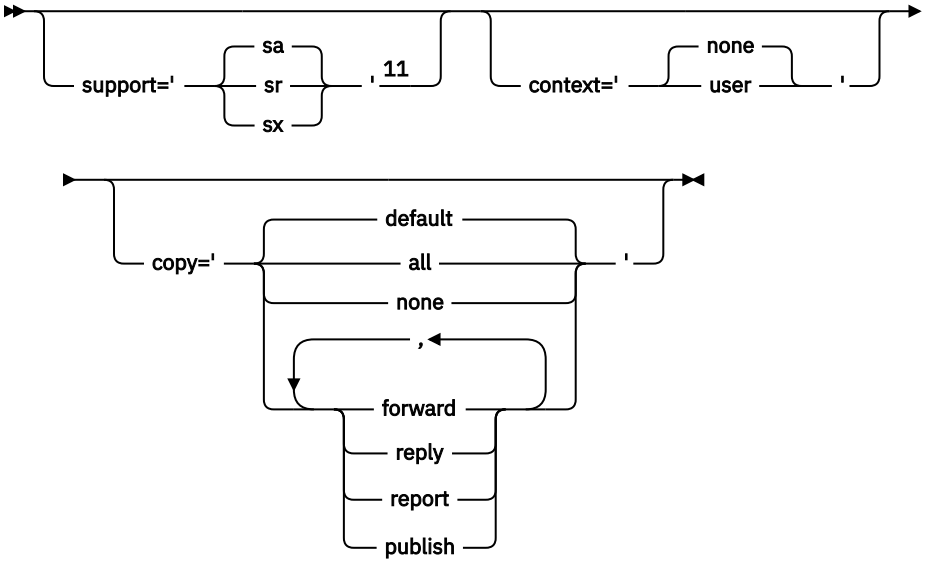
Właściwości



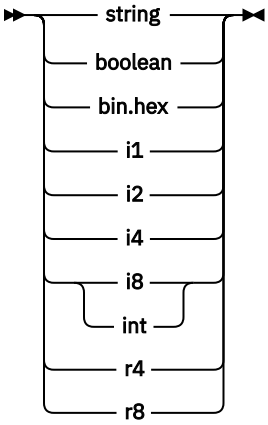
Właściwość



Atrybut właściwości



Typy danych



Uwagi:

[1] Znaki podwójnego cudzysłowu lub pojedyncze cudzysłowy są poprawne.

[2] Nie należy używać niepoprawnej nazwy właściwości; patrz sekcja “Niepoprawna nazwa właściwości” na stronie 564. Zastrzeżonej nazwy właściwości należy używać tylko w zdefiniowanym przeznaczeniu. Informacje na ten temat zawiera sekcja “Zdefiniowane nazwy właściwości” na stronie 564.

[3] Nazwa musi być zapisana małymi literami.

[4] Obsługiwany jest tylko jeden folder `psc` i `pscr` .

[5] Produkt WebSphere Application Server Service Integration Bus ignoruje foldery sib, sib_contexti sib_usr w kolejnych nagłówkach MQRFH2 , a tylko właściwości w pierwszym nagłówku MQRFH2 są istotne.

[6] W MQRFH2musi być obecny nie więcej niż jeden folder `usr` . Właściwości w folderze `usr` nie mogą występować więcej niż jeden raz.

[7] Istotne są tylko właściwości w pierwszym folderze `mq` . Jeśli folder to UTF-8, obsługiwane są tylko znaki jednobajtowe UTF-8 . Jedynym białym znakiem jest Unicode U+0020.

[8] Poprawne znaki są zdefiniowane w specyfikacji XML W3C i składają się zasadniczo z kategorii Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` oraz Nd ; patrz Kategorie znaków Unicode.

[9] Wszystkie znaki są istotne. Odstępy wiodące i końcowe są częścią wartości elementu.

[10] Nie należy używać niepoprawnego znaku; patrz “Nieprawidłowe znaki” na stronie 564. Użyj sekwencji o zmienionym znaczeniu, a nie tych niepoprawnych znaków.

[11] Atrybut właściwości obsługi jest poprawny tylko w folderze `mq` .

Nazwa folderu

Plik *NameValueData* zawiera pojedynczy folder. Aby utworzyć wiele folderów, należy utworzyć wiele pól *NameValueData* . W jednym nagłówku MQRFH2 komunikatu można utworzyć wiele pól *NameValueData* . Można również utworzyć wiele połączonych nagłówków MQRFH2 , z których każdy zawiera wiele pól *NameValueData* .

Kolejność nagłówków MQRFH2 i kolejność pól *NameValueData* nie ma znaczenia dla logicznej zawartości folderu. Jeśli ten sam folder występuje więcej niż jeden raz w komunikacie, folder jest analizowany jako całość. Jeśli ta sama właściwość występuje w wielu instancjach tego samego folderu, jest analizowana jako lista.

Na poprawną analizę składni elementu MQRFH2 nie mają wpływu alternatywne sposoby fizycznego zapisywania folderu w komunikacie.

Cztery foldery nie są zgodne z tą regułą. Analizowana jest tylko pierwsza instancja folderu `mq`, `sib`, `sib_contexti` `sib_usr` .

Jeśli ta sama właściwość występuje więcej niż jeden raz w połączonej treści połączonych nagłówków MQRFH2 , analizowana jest tylko pierwsza instancja właściwości. Jeśli właściwość jest ustawiana za pomocą wywołania interfejsu API, takiego jak MQSETMP, i dodawana do MQRFH2 bezpośrednio przez aplikację, pierwszeństwo ma wywołanie interfejsu API.

Nazwa folderu jest nazwą folderu zawierającego pary nazwa-wartość lub grupy. Grupy i pary nazwa-wartość mogą być mieszane na tym samym poziomie w drzewie folderów; patrz sekcja Rysunek 1 na stronie 553. Nie należy łączyć nazwy grupy i nazwy elementu; patrz sekcja Rysunek 2 na stronie 554

```
<group1><nvp1>value</nvp1></group1><group2><nvp2>value</nvp2></group2>
<group3><nvp1>value</nvp1></group3><nvp3>value</nvp3>
```

Rysunek 1. Poprawne użycie grup i par nazwa-wartość

```
<group1><nvp1> value </nvp1> value </group1>
```

Rysunek 2. Niepoprawne użycie grup i par nazwa-wartość

Nie należy używać niepoprawnej lub zarezerwowanej nazwy folderu; patrz “Niepoprawna nazwa ścieżki” na stronie 563 i “Nazwa zarezerwowanego folderu lub folderu właściwości” na stronie 563. Zdefiniowanej nazwy folderu należy używać tylko w celu jej zdefiniowania; patrz sekcja “Nazwa zdefiniowanego folderu” na stronie 555.

Jeśli atrybut `'content=properties'` zostanie dodany do znacznika nazwy folderu, folder stanie się folderem właściwości; patrz sekcja Rysunek 3 na stronie 554.

```
<myFolder></myfolder>
<myPropertyFolder contents='properties'></myPropertyFolder>
```

Rysunek 3. Przykład folderu i folderu właściwości

W nazwach folderów rozróżniana jest wielkość liter. Nazwy folderów i nazwy folderów właściwości współużytkują tę samą przestrzeń nazw. Muszą mieć różne nazwy. `Folder1` w Rysunek 4 na stronie 554 musi mieć inną nazwę niż `Folder2` w Rysunek 5 na stronie 554.

```
< Folder1 ><NVP1> value </NVP1></ Folder1 >
```

Rysunek 4. `Folder1` przestrzeń nazw

```
< Folder2 content='properties'>< Property1 > value </ Property1 ></ Folder2 >
```

Rysunek 5. `Folder2` przestrzeń nazw

Grupy, właściwości i pary nazwa-wartość w różnych folderach mają różne przestrzenie nazw. `Property1` w Rysunek 5 na stronie 554 jest inną właściwością niż `Property1` w Rysunek 6 na stronie 554.

```
<Folder3 content='properties'>< Property1 > value </ Property1 ></Folder3>
```

Rysunek 6. `Folder3` przestrzeń nazw

Foldery właściwości różnią się od folderów innych niż foldery właściwości pod dwoma ważnymi względami:

1. Foldery właściwości zawierają właściwości, a foldery inne niż właściwości zawierają pary nazwa-wartość. Foldery różnią się nieznacznie, pod względem składniowym.
2. Aby uzyskać dostęp do właściwości komunikatu, należy użyć zdefiniowanych interfejsów, takich jak właściwości MQI lub właściwości komunikatu JMS . Interfejsy zapewniają, że foldery właściwości w pliku `MQRFH2` są poprawnie sformatowane. Poprawnie sformatowany folder właściwości współdziała między menedżerami kolejek na różnych platformach i w różnych wersjach.

Właściwość komunikatu MQI jest stabilnym sposobem odczytywania i zapisywania strumienia MQRFH2i pozwala uniknąć trudności z poprawnym analizowaniem strumienia MQRFH2 .

Nazwa zdefiniowanego folderu

Zdefiniowana nazwa folderu jest nazwą folderu, która jest zarezerwowana dla produktu IBM MQlub innego produktu. Nie należy tworzyć folderu o takiej samej nazwie i nie dodawać do folderów własnych par nazwa-wartość. Zdefiniowane foldery to psc i pscr.

Produkt psc i produkt pscr są używane przez umieszczoną w kolejce publikację/subskrypcję.

Segmentowany komunikat umieszczony za pomocą elementu MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED nie może zawierać elementu MQRFH2 ze zdefiniowaną nazwą folderu. Operacja MQPUT kończy się niepowodzeniem z kodem przyczyny 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nazwa zdefiniowanego folderu właściwości

Zdefiniowana nazwa folderu właściwości to nazwa folderu właściwości używanego przez produkt IBM MQlub inny produkt. Nazwy folderów i ich treść zawiera sekcja Foldery właściwości. Zdefiniowane nazwy folderów właściwości są podzbiorem wszystkich nazw folderów zarezerwowanych przez produkt IBM MQ; patrz sekcja “Nazwa zarezerwowanego folderu lub folderu właściwości” na stronie 563.

Każdy element przechowywany w zdefiniowanym folderze właściwości jest właściwością. Element przechowywany w zdefiniowanym folderze właściwości nie może mieć atrybutu content='properties' .

Właściwości można dodawać tylko do zdefiniowanych folderów właściwości usr, mq_usri sib_usr. W innych folderach właściwości, takich jak mq i sib, produkt IBM MQ ignoruje lub odrzuca właściwości, których nie rozpoznaje.

Opis każdego zdefiniowanego folderu właściwości zawiera listę zdefiniowanych przez IBM MQ właściwości, które mogą być używane przez aplikacje. Dostęp do niektórych właściwości można uzyskać pośrednio, ustawiając lub pobierając właściwość JMS , a dostęp do niektórych właściwości można uzyskać bezpośrednio przy użyciu wywołań MQI produktu MQSETMP i MQINQMP .

Zdefiniowane foldery właściwości zawierają również inne właściwości, które zostały zarezerwowane przez produkt IBM MQ , ale do których aplikacje nie mają dostępu. Nazwy zarezerwowanych właściwości nie są wyświetlane. W folderach właściwości usr, mq_usri sib_usr nie ma zarezerwowanych właściwości. Nie należy jednak tworzyć właściwości z niepoprawnymi nazwami właściwości; patrz sekcja “Niepoprawna nazwa właściwości” na stronie 564.

Foldery właściwości

jms

jms zawiera pola nagłówka JMS oraz właściwości JMSX, które nie mogą być w pełni wyrażone w produkcie MQMD. Folder jms jest zawsze obecny w produkcie MQRFH2 usługi Java Message Service.

Tabela 515. Nazwa właściwości jms, synonim, typ danych i folder

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
JMSDestination	jms.Dst	string	<jms><Dst> *destination* </Dst></jms>
JMSExpiration	jms.Exp	i8	<jms><Exp> *expiration* </Exp></jms>
JMSCorrelation	jms.Cid	string	<jms><Cid> *correlationId* </Cid></jms>

Tabela 515. Nazwa właściwości jms, synonim, typ danych i folder (kontynuacja)			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
Dostarczanie JMS	jms.Dlv	i4	<jms><Dlv> *delivery* </Dlv></jms>
JMSPriority	jms.Pri	i4	<jms><Pri> *priority* </Pri></jms>
JMSReplyTo	jms.Rto	string	<jms><Rto> *replyToURI* </Rto></jms>
JMSTimeStamp	jms.Tms	i8	<jms><Tms> *timestamp* </Tms></jms>
JMSXGroupID	jms.Gid	string	<jms><Gid> *groupId* </Gid></jms>
JMSXGroupSeq	jms.Seq	i4	<jms><Seq> *messageSequenceNo* </Seq></jms>

Nie należy dodawać własnych właściwości w folderze jms.

mcd

mcd zawiera właściwości opisujące format komunikatu. Na przykład właściwość domeny usługi komunikatu Msd identyfikuje komunikat JMS jako JMSTextMessage, JMSBytesMessage, JMSStreamMessage, JMSMapMessage, JMSObjectMessage lub wartość NULL.

Folder mcd jest zawsze obecny w komunikacie usługi Java Message Service zawierającym MQRFH2.

Jest on zawsze obecny w komunikacie zawierającym element MQRFH2 wysłanym z programu IBM Integration Bus. Opisuje on domenę, format, typ i zestaw komunikatu.

Tabela 516. mcd - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
	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>

Nie należy dodawać własnych właściwości w folderze mcd.

mq_usr

Plik mq_usr zawiera właściwości zdefiniowane przez aplikację, które nie są prezentowane jako właściwości zdefiniowane przez użytkownika JMS . W tym folderze można umieścić właściwości, które nie spełniają wymagań JMS .

Właściwości można tworzyć w folderze mq_usr . Właściwości utworzone w mq_usr są podobne do właściwości utworzonych w nowych folderach z atrybutem content='properties' .

sib

Plik sib zawiera właściwości komunikatu systemowego magistrali integracji usług (WAS/SIB) systemu WebSphere Application Server . Właściwości sib nie są prezentowane jako właściwości JMS dla aplikacji IBM MQ JMS , ponieważ nie są obsługiwane. Na przykład niektóre właściwości sib nie mogą być prezentowane jako właściwości JMS , ponieważ są tablicami bajtów. Niektóre właściwości sib są ujawniane aplikacjom WAS/SIB jako właściwości JMS_IBM_* . Są to między innymi właściwości ścieżek routingu skierowanego do przodu i routingu zwrotnego.

Nie należy dodawać własnych właściwości w folderze sib.

sib_context

Plik sib_context zawiera właściwości komunikatu systemowego WAS/SIB, które nie są ujawniane aplikacjom użytkownika WAS/SIB ani jako właściwości JMS . Plik sib_context zawiera właściwości zabezpieczeń i właściwości transakcyjne używane na potrzeby usług Web Service.

Nie należy dodawać własnych właściwości w folderze sib_context.

sib_usr

Plik sib_usr zawiera właściwości komunikatu użytkownika WAS/SIB, które nie są prezentowane jako właściwości użytkownika JMS , ponieważ nie są obsługiwane. Produkt sib_usr jest ujawniany dla aplikacji WAS/SIB w interfejsie produktu SIMessage . Więcej informacji na ten temat zawiera sekcja Projektowanie integracji usług.

Właściwość sib_usr musi mieć typ bin.hex, a wartość musi mieć poprawny format. Jeśli aplikacja IBM MQ zapisuje element o typie bin.hex w folderze w niewłaściwym formacie, otrzyma ona element IOException. Jeśli typem danych właściwości nie jest bin.hex , aplikacja odbiera ClassCastException.

Nie należy podejmować próby udostępnienia właściwości użytkownika JMS w systemie WAS/SIB przy użyciu tego folderu. Zamiast tego należy użyć folderu usr .

Właściwości można tworzyć w folderze sib_usr .

usr

usr zawiera zdefiniowane przez aplikację właściwości JMS powiązane z komunikatem. Folder usr jest obecny tylko wtedy, gdy w aplikacji ustawiono właściwość definiowaną przez aplikację.

usr jest domyślnym folderem właściwości. Jeśli właściwość jest ustawiona bez nazwy folderu, jest ona umieszczana w folderze usr .

Tabela 517. nazwa właściwości usr, synonim, typ danych i folder

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	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>

Właściwości można tworzyć w folderze usr .

Segmentowany komunikat umieszczony za pomocą elementu MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED nie może zawierać elementu MQRFH2 ze zdefiniowaną nazwą folderu właściwości. Operacja MQPUT kończy się niepowodzeniem z kodem przyczyny 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Nazwa niezgrupowanego folderu właściwości

ibm

ibm zawiera właściwości, które są używane tylko przez IBM MQ.

Tabela 518. ibm - nazwa właściwości, synonim, typ danych i folder

Synonim właściwości	Nazwa właściwości	Typ danych	Folder
	ibm.rfp	string	<ibm><rfp>fingerprint</rfp></ibm>

Nie należy dodawać własnych właściwości w folderze ibm.

mq

mq zawiera właściwości, które są używane tylko przez IBM MQ.

Do właściwości w folderze mq mają zastosowanie następujące ograniczenia:

- Tylko właściwości w pierwszym znaczącym folderze mq w komunikacie są uwzględniane przez produkt MQ. Właściwości w każdym innym folderze mq w komunikacie są ignorowane.
- W folderze dozwolone są tylko jednobajtowe znaki UTF-8 . Wielobajtowy znak w folderze może spowodować niepowodzenie analizowania i odrzucenie komunikatu.
- W folderze nie należy używać łańcuchów o zmienionym znaczeniu. Łańcuch o zmienionym znaczeniu jest traktowany jako rzeczywista wartość elementu.
- Tylko znaki Unicode U+0020 są traktowane jako białe znaki w folderze. Wszystkie pozostałe znaki są traktowane jako istotne i mogą spowodować niepowodzenie analizowania folderu oraz odrzucenie komunikatu.

Jeśli analizowanie folderu mq nie powiedzie się lub jeśli folder nie przestrzega tych ograniczeń, komunikat zostanie odrzucony z kodem przyczyny 2527, MQRC_RFH_RESTRICTED_FORMAT_ERR.

Nie należy dodawać własnych właściwości w folderze mq.

mqema

mqema zawiera właściwości, które są używane tylko przez WebSphere Application Server. Folder został zastąpiony przez folder mqext.

Nie należy dodawać własnych właściwości w folderze mqema.

mqext

Produkt mqext zawiera następujące typy właściwości:

- Właściwości, które są używane tylko przez produkt WebSphere Application Server.
- Właściwości związane z opóźnionym dostarczaniem komunikatów.

Folder jest obecny, jeśli w przypadku aplikacji ustawiono co najmniej jedną właściwość zdefiniowaną przez IBM albo używane jest opóźnienie dostawy.

Tabela 519. mqext - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
JMSArmCorrelator	mqext.Arm	string	<mqext><Arm>*armCorrelator*</Arm></mqext>
JMSRMCorrelator	mqext.Wrm	string	<mqext><Wrm>*wrmCorrelator*</Wrm></mqext>
JMSDeliveryTime	mqext.Dlt	i8	<mqext><Dlt>*DeliveryTime*</Dlt></mqext>
JMSDeliveryDelay	mqext.Dly	i8	<mqext><Dly>*DeliveryTime*</Dly></mqext>

Nie należy dodawać własnych właściwości w folderze mqext.

mqps

mqps zawiera właściwości, które są używane tylko przez proces publikowania/subskrybowania produktu IBM MQ. Folder jest obecny tylko wtedy, gdy w przypadku aplikacji ustawiono co najmniej jedną zintegrowaną właściwość publikowania/subskrybowania.

Tabela 520. mqps - nazwa właściwości, synonim, typ danych i folder			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
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>

Nie należy dodawać własnych właściwości w folderze mqps.

mq_svc

Plik mq_svc zawiera właściwości używane przez pakiet SupportPac MA93.

Nie należy dodawać własnych właściwości w folderze mq_svc.

mqtt

mqtt zawiera właściwości używane przez MQ Telemetry

Tabela 521. Nazwa właściwości mqtt, synonim, typ danych i folder			
Synonim właściwości	**Nazwa właściwości**	**Typ danych**	**Folder**
	mqtt.clientId	string	<mqtt><clientId> *topicString* </clientId></mqtt>
	mqtt.qos	i4	<mqtt><qos> *qualityOfService* </qos></mqtt>
	mqtt.msgid	string	<mqtt><msgid> *messageIdentifier* </msgid></mqtt>

Nie należy dodawać własnych właściwości w folderze mqtt.

Segmentowany komunikat umieszczony za pomocą elementu MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED nie może zawierać elementu MQRFH2 z nazwą folderu właściwości rozgrupowanej. Operacja MQPUT kończy się niepowodzeniem z kodem przyczyny 2443, MQRC_SEGMENTATION_NOT_ALLOWED.

Pary nazwa-wartość

Na diagramie składniowym "Pary nazwa-wartość" opisuje zawartość zwykłego folderu. Zwykły folder zawiera grupy i składniki. Element jest parą nazwa-wartość. Grupa zawiera elementy i inne grupy.

Jeśli chodzi o drzewa, elementy są węzłami liści, a grupy są węzłami wewnętrznymi. Węzeł wewnętrzny i folder, który jest węzłem głównym, mogą zawierać zarówno węzły wewnętrzne, jak i węzły-liście. Węzeł nie może być jednocześnie węzłem wewnętrznym i węzeł-liściem; patrz Rysunek 2 na stronie 554.

Właściwości

Na diagramie składni sekcja "Właściwości" opisuje zawartość folderu właściwości. Folder właściwości zawiera grupy i właściwości. Właściwość jest parą nazwa-wartość z opcjonalnym atrybutem typu danych. Grupa zawiera właściwości i inne grupy.

Jeśli chodzi o drzewa, właściwości są węzłami liści, a grupy są węzłami wewnętrznymi. Węzeł wewnętrzny i folder właściwości, który jest węzłem głównym, mogą zawierać zarówno węzły wewnętrzne, jak i węzły-liście. Węzeł nie może być jednocześnie węzłem wewnętrznym i węzeł-liściem; patrz Rysunek 2 na stronie 554.

Właściwość

Właściwość komunikatu jest parą nazwa-wartość w folderze właściwości. Opcjonalnie może zawierać atrybut typu danych i atrybut właściwości. Na przykład można zapoznać się z poniższym kodem. Jeśli atrybut typu danych zostanie pominięty, właściwość będzie miała typ string.

```
<pf><p1 dt='i8' > value </p1></pf>
```

Nazwa właściwości komunikatu jest pełną nazwą ścieżki z podobną do XML składnią <> , zastąpioną kropkami. Na przykład łańcuch myPropertyFolder1.myGroup1.myGroup2.myProperty1 jest odwzorowywany na łańcuch *NameValueData* w następujący sposób. Łańcuch jest sformatowany w celu łatwiejszego odczytu.

```
<myPropertyFolder1>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
        </myGroup2>
    </myGroup1>
</myPropertyFolder1>
```

Folder właściwości może zawierać wiele właściwości. Na przykład właściwości w pliku Rysunek 7 na stronie 561 są odwzorowywane na folder właściwości w pliku Zmiany w celu odizolowania kolejki sprzedaży w nowym klastrze i oddzielenia kolejek transmisji klastra bramy.

```
 myPropertyFolder1.myProperty4
myPropertyFolder1.myGroup1.myGroup2.myProperty1
myPropertyFolder1.myGroup1.myGroup2.myProperty2
myPropertyFolder1.myGroup1.myProperty3
```

Rysunek 7. Wiele właściwości o tej samej nazwie głównej

```
<myPropertyFolder1>
    <myProperty4>value</myProperty4>
    <myGroup1>
        <myGroup2>
            <myProperty1>value</myProperty1>
            <myProperty2>value</myProperty2>
        </myGroup2>
        <myProperty3>value</myProperty3>
    </myGroup1>
</myPropertyFolder1>
```

Rysunek 8. Odwzorowanie wielu nazw właściwości

Nazwa

Nazwa musi zaczynać się od *List* lub *Podkreślenie*. Nie może zawierać *dwukropka*, nie może kończyć się *kropką* i zawierać tylko *liter*, *cyfr*, *znaków podkreślenia*, *łączników*i *kropek*. Poprawne znaki są zdefiniowane w specyfikacji XML W3C i składają się zasadniczo z kategorii Unicode `Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm,` oraz `Nd` ; patrz Kategorie znaków Unicode.

Pełna ścieżka do właściwości lub pary nazwa-wartość nie może naruszać reguły opisanej w sekcji “Niepoprawna nazwa ścieżki” na stronie 563. Długość ścieżek jest ograniczona do 4095 bajtów, ścieżki nie mogą zawierać znaków zgodności ze standardem Unicode i nie mogą zaczynać się od łańcucha XML.

Nazwa grupy

Nazwa grupy ma taką samą składnię jak nazwa. Nazwy grup są opcjonalne. Właściwości i pary nazwa-wartość można umieścić w katalogu głównym folderu. Użyj grup, jeśli pomaga to zorganizować właściwości i pary nazwa-wartość.

Nazwa elementu

Nazwa elementu ma taką samą składnię jak nazwa.

Wartość elementu

Wartość elementu zawiera wszystkie białe znaki między znacznikiem < *Element name* > i elementem < */Element name* >. W wartości nie należy używać dwóch znaków < i & . Następnie należy zastąpić < i `&`.

Atrybuty elementu Property

Pola deskryptora właściwości odwzorowania atrybutów właściwości: Odwzorowania są następujące:

Obsługa

sa (wartość domyślna)
MQPD_SUPPORT_OPTIONAL

SR
MQPD_SUPPORT_REQUIRED

sx
MQPD_SUPPORT_REQUIRED_IF_LOCAL

Kontekst

none (brak) (wartość domyślna)
MQPD_NO_CONTEXT

użytkownik
MQPD_USER _CONTEXT

CopyOptions

postępująca
MQPD_COPY_FORWARD

reply
MQPD_COPY_REPLY

raportowanie
MQPD_COPY_REPORT

publikować
MQPD_COPY_PUBLISH

wszystkie
MQPD_COPY_ALL

Nie należy używać opcji `all` w połączeniu z innymi opcjami.

default
MQPD_COPY_DEFAULT

Nie należy używać opcji `default` w połączeniu z innymi opcjami. Wartość `default` jest taka sama jak `forward` + `report` + `publish`.

brak
MQPD_COPY_NONE

Nie należy używać wartości `none` w połączeniu z innymi opcjami.

Atrybuty właściwości Support mają zastosowanie tylko do właściwości w folderze `mq` .

Atrybuty właściwości Context i CopyOptions mają zastosowanie do wszystkich folderów właściwości.

Typy danych

Typy danych `MQRFH2` są odwzorowywane na typy właściwości komunikatu w następujący sposób:

Tabela 522. Odwzorowania typów danych

MQRFH2 Typ danych	**Typ właściwości komunikatu**
`bin.hex`	`MQBYTE[]`
`boolean`	`MQBOOL`
`i1`	`MQINT8`
`i2`	`MQINT16`
`i4`	`MQINT32`

Tabela 522. Odwzorowania typów danych (kontynuacja)	
MQRFH2 Typ danych	**Typ właściwości komunikatu**
i8	MQINT64
r4	MQFLOAT32
r8	MQFLOAT64
string	MQCHAR[]

Przyjmuje się, że każdy element bez typu danych ma typ `string`.

Wartość NULL jest wskazywana przez atrybut elementu `xsi:nil='true'`. Nie należy używać atrybutu `xsi:nil='false'` dla wartości innych niż NULL. Na przykład następująca właściwość ma wartość null:

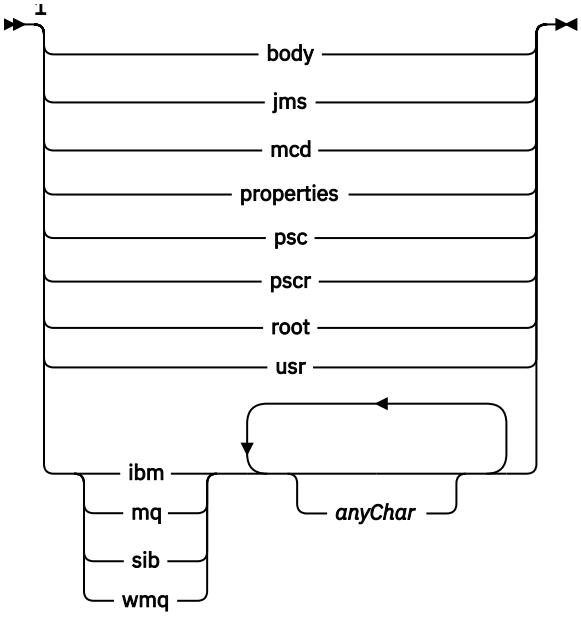
```
<NullProperty
xsi:nil='true'></NullProperty>
```

Właściwość bajtu lub łańcucha znaków może mieć pustą wartość. Wartość pusta jest reprezentowana przez element MQRFH2 z wartością elementu o zerowej długości. Na przykład następująca właściwość ma pustą wartość:

```
<EmptyProperty></EmptyProperty>
```

Nazwa zarezerwowanego folderu lub folderu właściwości

Ogranicz nazwę folderu lub folderu właściwości tak, aby nie rozpoczynała się od żadnego z następujących łańcuchów. Przedrostki są zarezerwowane dla nazw folderów lub właściwości utworzonych przez IBM.

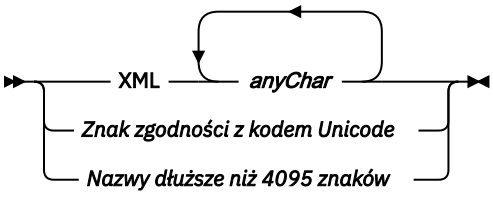


Uwagi:

[1] Zastrzeżona nazwa folderu lub właściwości zawiera dowolną kombinację małych i wielkich liter.

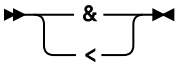
Niepoprawna nazwa ścieżki

Ogranicz pełną ścieżkę do pary nazwa-wartość lub właściwości, aby nie zawierała żadnego z następujących łańcuchów.



Nieprawidłowe znaki

Zawsze należy używać sekwencji o zmienionym znaczeniu & i < zamiast literałów "&" i "<".

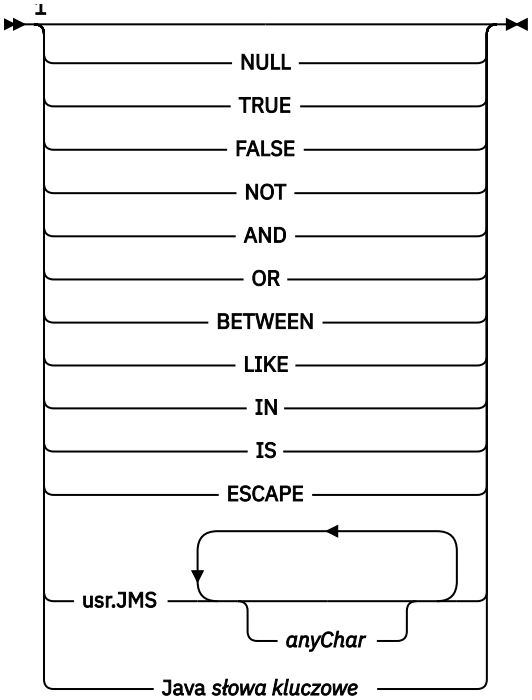


Zdefiniowane nazwy właściwości

Zdefiniowane nazwy właściwości to nazwy właściwości, które są definiowane przez produkt IBM MQlub inne produkty i używane przez aplikacje IBM MQ i aplikacje użytkownika. Zdefiniowane właściwości istnieją tylko w zdefiniowanych folderach właściwości. Zdefiniowane nazwy właściwości są opisane w opisie folderów właściwości; patrz sekcja Foldery właściwości.

Niepoprawna nazwa właściwości

Nie należy tworzyć nazw właściwości zgodnych z następującą regułą. Reguła ma zastosowanie do pełnej ścieżki właściwości, która określa właściwość, a nie tylko do nazwy elementu właściwości.



Uwagi:

[1] Niepoprawna nazwa właściwości może zawierać dowolną kombinację wielkich i małych liter.

Niepoprawne atrybuty

Foldery i właściwości właściwości mogą zawierać tylko obsługiwane “Atrybuty elementu Property” na stronie 561 i “Typy danych” na stronie 562.

Wszystkie nieobsługiwane atrybuty podobne do XML, na przykład nazwy z cytowanymi wartościami łańcuchowymi, które są zawarte w folderach właściwości lub właściwościach, mogą zostać usunięte.

Atrybuty podobne do XML zawarte w folderach innych niż właściwości lub w elementach innych niż właściwości, które pozostają w nagłówkach MQRFH2 .

MQRMH-nagłówek komunikatu referencyjnego

Struktura MQRMH definiuje format nagłówka komunikatu odniesienia. Ten nagłówek jest używany z wyjściami kanału komunikatów zapisanymi przez użytkownika do wysyłania bardzo dużych ilości danych (nazywanych *danymi masowymi*). z jednego menedżera kolejek do innego. Różnica w porównaniu do normalnego przesyłania komunikatów polega na tym, że dane masowe nie są przechowywane w kolejce. Zamiast tego w kolejce zapisywane jest tylko *odwołanie* do danych masowych. Zmniejsza to prawdopodobieństwo wyczerpania zasobów IBM MQ przez niewielką liczbę bardzo dużych komunikatów.

Dostępność

Struktura MQRMH jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Nazwa formatu

MQFMT_REF_MSG_HEADER

Zestaw znaków i kodowanie

Dane znakowe w programie MQRMH oraz łańcuchy adresowane przez pola przesunięcia muszą znajdować się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek systemu **CodedCharSetId** . Dane liczbowe w programie MQRMH muszą być zakodowane na komputerze rodzimym; jest to podawane przez wartość parametru MQENC_NATIVE dla języka programowania C.

Ustaw zestaw znaków i kodowanie MQRMH w polach *CodedCharSetId* i *Encoding* w następujących polach:

- MQMD (jeśli struktura MQRMH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQRMH (wszystkie inne przypadki).

Użycie

Aplikacja umieszcza komunikat składający się z MQRMH, ale pomija dane masowe. Gdy agent kanału komunikatów (MCA) odczytuje komunikat z kolejki transmisji, wywoływana jest procedura zewnętrzna dostarczona przez użytkownika w celu przetworzenia nagłówka komunikatu odniesienia. Wyjście może dodać do komunikatu odwołania dane masowe identyfikowane przez strukturę MQRMH, zanim agent MCA wyśle komunikat za pośrednictwem kanału do następnego menedżera kolejek.

Na odbierającym końcu musi istnieć wyjście komunikatu, które oczekuje na komunikaty odniesienia. Po odebraniu komunikatu odniesienia wyjście musi utworzyć obiekt na podstawie danych masowych, które następują po komunikacie MQRMH, a następnie przekazać komunikat odniesienia bez danych masowych. Komunikat odniesienia może zostać później pobrany przez aplikację, która odczytuje komunikat odniesienia (bez danych masowych) z kolejki.

Zwykle struktura MQRMH jest wszystkim, co znajduje się w komunikacie. Jeśli jednak komunikat znajduje się w kolejce transmisji, jeden lub więcej dodatkowych nagłówków poprzedza strukturę MQRMH.

Komunikat odniesienia może być również wysłany do listy dystrybucyjnej. W tym przypadku struktura MQDH i powiązane z nią rekordy poprzedzają strukturę MQRMH, gdy komunikat znajduje się w kolejce transmisji.

Uwaga: Nie należy wysyłać komunikatu odniesienia jako komunikatu segmentowanego, ponieważ wyjście komunikatu nie może go poprawnie przetworzyć.

Konwersja danych

Na potrzeby konwersji danych struktura MQRMH obejmuje konwersję danych środowiska źródłowego, nazwy obiektu źródłowego, danych środowiska docelowego i nazwy obiektu docelowego. Wszystkie inne bajty w obrębie *StrucLength* bajtów początku struktury są odrzucane lub mają niezdefiniowane wartości po konwersji danych. Dane masowe są przekształcane pod warunkiem, że wszystkie poniższe stwierdzenia są prawdziwe:

- Dane masowe są obecne w komunikacie podczas wykonywania konwersji danych.
- Pole *Format* w MQRMH ma wartość inną niż MQFMT_NONE.
- Istnieje zapisane przez użytkownika wyjście konwersji danych o podanej nazwie formatu.

Należy jednak pamiętać, że zwykle dane masowe nie są obecne w komunikacie, gdy komunikat znajduje się w kolejce, a w wyniku tego dane masowe są przekształcane za pomocą opcji MQGMO_CONVERT.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 523. Pola w MQRMH dla MQRMH

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQRMH_STRUC_ID (ID struktury MQRM)	'RMH¬'
Wersja (numer wersji struktury)	MQRMH_VERSION_1	1
StrucLength (łączna długość MQRMH, w tym łańcuchy na końcu pól stałych, ale nie dane masowe)	Brak	0
Kodowanie (kodowanie liczbowe dla danych masowych)	RODZIMA MQENC	Zależy od środowiska
CodedCharSetId (identyfikator zestawu znaków danych masowych)	MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0
Format (nazwa formatu danych masowych)	MQFMT_BRAK	Puste
Flagi (patrz flagi komunikatów)	MQRMHF_NOT_LAST (nie)	0
ObjectType (typ obiektu)	Brak	Puste
ObjectInstanceId (identyfikator instancji obiektu)	MQOII_BRAK	Wartości null
SrcEnvDługość (długość danych środowiska źródłowego)	Brak	0

Tabela 523. Pola w MQRMH dla MQRMH (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
SrcEnvPrzesunięcie (przesunięcie danych środowiska źródłowego)	Brak	0
SrcNameDługość (długość nazwy obiektu źródłowego)	Brak	0
SrcNamePrzesunięcie (przesunięcie nazwy obiektu źródłowego)	Brak	0
DestEnvDługość (długość danych środowiska docelowego)	Brak	0
DestEnvPrzesunięcie (przesunięcie danych środowiska docelowego)	Brak	0
DestNameDługość (długość nazwy obiektu docelowego)	Brak	0
DestNamePrzesunięcie (przesunięcie nazwy obiektu docelowego)	Brak	0
DataLogicalLength (długość danych masowych)	Brak	0
DataLogicalOffset (niskie przesunięcie danych masowych)	Brak	0
DataLogicalOffset2 (duże przesunięcie danych masowych)	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQRMH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQRMH MyRMH = {MQRMH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQRMH

```
typedef struct tagMQRMH MQRMH;
struct tagMQRMH {
  MQCHAR4   StrucId;             /* Structure identifier */
  MQLONG    Version;             /* Structure version number */
  MQLONG    StrucLength;         /* Total length of MQRMH, including
                                    strings at end of fixed fields, but
                                    not the bulk data */
  MQLONG    Encoding;            /* Numeric encoding of bulk data */
  MQLONG    CodedCharSetId;      /* Character set identifier of bulk
                                    data */
  MQCHAR8   Format;              /* Format name of bulk data */
  MQLONG    Flags;               /* Reference message flags */
  MQCHAR8   ObjectType;          /* Object type */
  MQBYTE24  ObjectInstanceId;    /* Object instance identifier */
  MQLONG    SrcEnvLength;        /* Length of source environment data */
  MQLONG    SrcEnvOffset;        /* Offset of source environment data */
  MQLONG    SrcNameLength;       /* Length of source object name */
  MQLONG    SrcNameOffset;       /* Offset of source object name */
  MQLONG    DestEnvLength;       /* Length of destination environment
                                    data */
```

```
  MQLONG    DestEnvOffset;       /* Offset of destination environment
                                    data */
  MQLONG    DestNameLength;      /* Length of destination object name */
  MQLONG    DestNameOffset;      /* Offset of destination object name */
  MQLONG    DataLogicalLength;   /* Length of bulk data */
  MQLONG    DataLogicalOffset;   /* Low offset of bulk data */
  MQLONG    DataLogicalOffset2;  /* High offset of bulk data */
};
```

Deklaracja języka COBOL dla MQRMH

```
**   MQRMH structure
  10 MQRMH.
**    Structure identifier
   15 MQRMH-STRUCID            PIC X(4).
**    Structure version number
   15 MQRMH-VERSION            PIC S9(9) BINARY.
**    Total length of MQRMH, including strings at end of fixed fields,
**    but not the bulk data
   15 MQRMH-STRUCLENGTH        PIC S9(9) BINARY.
**    Numeric encoding of bulk data
   15 MQRMH-ENCODING           PIC S9(9) BINARY.
**    Character set identifier of bulk data
   15 MQRMH-CODEDCHARSETID     PIC S9(9) BINARY.
**    Format name of bulk data
   15 MQRMH-FORMAT             PIC X(8).
**    Reference message flags
   15 MQRMH-FLAGS              PIC S9(9) BINARY.
**    Object type
   15 MQRMH-OBJECTTYPE         PIC X(8).
**    Object instance identifier
   15 MQRMH-OBJECTINSTANCEID   PIC X(24).
**    Length of source environment data
   15 MQRMH-SRCENVLENGTH       PIC S9(9) BINARY.
**    Offset of source environment data
   15 MQRMH-SRCENVOFFSET       PIC S9(9) BINARY.
**    Length of source object name
   15 MQRMH-SRCNAMELENGTH      PIC S9(9) BINARY.
**    Offset of source object name
   15 MQRMH-SRCNAMEOFFSET      PIC S9(9) BINARY.
**    Length of destination environment data
   15 MQRMH-DESTENVLENGTH      PIC S9(9) BINARY.
**    Offset of destination environment data
   15 MQRMH-DESTENVOFFSET      PIC S9(9) BINARY.
**    Length of destination object name
   15 MQRMH-DESTNAMELENGTH     PIC S9(9) BINARY.
**    Offset of destination object name
   15 MQRMH-DESTNAMEOFFSET     PIC S9(9) BINARY.
**    Length of bulk data
   15 MQRMH-DATALOGICALLENGTH  PIC S9(9) BINARY.
**    Low offset of bulk data
   15 MQRMH-DATALOGICALOFFSET  PIC S9(9) BINARY.
**    High offset of bulk data
   15 MQRMH-DATALOGICALOFFSET2 PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla MQRMH

```
dcl
 1 MQRMH based,
  3 StrucId            char(4),       /* Structure identifier */
  3 Version            fixed bin(31), /* Structure version number */
  3 StrucLength        fixed bin(31), /* Total length of MQRMH,
                                         including strings at end of
                                         fixed fields, but not the bulk
                                         data */
  3 Encoding           fixed bin(31), /* Numeric encoding of bulk
                                         data */
  3 CodedCharSetId     fixed bin(31), /* Character set identifier of
                                         bulk data */
  3 Format             char(8),       /* Format name of bulk data */
  3 Flags              fixed bin(31), /* Reference message flags */
  3 ObjectType         char(8),       /* Object type */
  3 ObjectInstanceId   char(24),      /* Object instance identifier */
  3 SrcEnvLength       fixed bin(31), /* Length of source environment
                                         data */
  3 SrcEnvOffset       fixed bin(31), /* Offset of source environment
                                         data */
```

```
  3 SrcNameLength      fixed bin(31), /* Length of source object name */
  3 SrcNameOffset      fixed bin(31), /* Offset of source object name */
  3 DestEnvLength      fixed bin(31), /* Length of destination
                                         environment data */
  3 DestEnvOffset      fixed bin(31), /* Offset of destination
                                         environment data */
  3 DestNameLength     fixed bin(31), /* Length of destination object
                                         name */
  3 DestNameOffset     fixed bin(31), /* Offset of destination object
                                         name */
  3 DataLogicalLength  fixed bin(31), /* Length of bulk data */
  3 DataLogicalOffset  fixed bin(31), /* Low offset of bulk data */
  3 DataLogicalOffset2 fixed bin(31); /* High offset of bulk data */
```

Deklaracja High Level Assembler dla MQRMH

```
MQRMH                     DSECT
MQRMH_STRUCID             DS   CL4   Structure identifier
MQRMH_VERSION             DS   F     Structure version number
MQRMH_STRUCLENGTH         DS   F     Total length of MQRMH, including
*                                     strings at end of fixed fields, but
*                                     not the bulk data
MQRMH_ENCODING            DS   F     Numeric encoding of bulk data
MQRMH_CODEDCHARSETID      DS   F     Character set identifier of bulk
*                                     data
MQRMH_FORMAT              DS   CL8   Format name of bulk data
MQRMH_FLAGS               DS   F     Reference message flags
MQRMH_OBJECTTYPE          DS   CL8   Object type
MQRMH_OBJECTINSTANCEID    DS   XL24  Object instance identifier
MQRMH_SRCENVLENGTH        DS   F     Length of source environment data
MQRMH_SRCENVOFFSET        DS   F     Offset of source environment data
MQRMH_SRCNAMELENGTH       DS   F     Length of source object name
MQRMH_SRCNAMEOFFSET       DS   F     Offset of source object name
MQRMH_DESTENVLENGTH       DS   F     Length of destination environment
*                                     data
MQRMH_DESTENVOFFSET       DS   F     Offset of destination environment
*                                     data
MQRMH_DESTNAMELENGTH      DS   F     Length of destination object name
MQRMH_DESTNAMEOFFSET      DS   F     Offset of destination object name
MQRMH_DATALOGICALLENGTH   DS   F     Length of bulk data
MQRMH_DATALOGICALOFFSET   DS   F     Low offset of bulk data
MQRMH_DATALOGICALOFFSET2  DS   F     High offset of bulk data
*
MQRMH_LENGTH              EQU  *-MQRMH
                          ORG  MQRMH
MQRMH_AREA                DS   CL(MQRMH_LENGTH)
```

Deklaracja Visual Basic dla MQRMH

```
Type MQRMH
  StrucId            As String*4 'Structure identifier'
  Version            As Long     'Structure version number'
  StrucLength        As Long     'Total length of MQRMH, including'
                                 'strings at end of fixed fields, but'
                                 'not the bulk data'
  Encoding           As Long     'Numeric encoding of bulk data'
  CodedCharSetId     As Long     'Character set identifier of bulk data'
  Format             As String*8 'Format name of bulk data'
  Flags              As Long     'Reference message flags'
  ObjectType         As String*8 'Object type'
  ObjectInstanceId   As MQBYTE24 'Object instance identifier'
  SrcEnvLength       As Long     'Length of source environment data'
  SrcEnvOffset       As Long     'Offset of source environment data'
  SrcNameLength      As Long     'Length of source object name'
  SrcNameOffset      As Long     'Offset of source object name'
  DestEnvLength      As Long     'Length of destination environment'
                                 'data'
  DestEnvOffset      As Long     'Offset of destination environment'
                                 'data'
  DestNameLength     As Long     'Length of destination object name'
  DestNameOffset     As Long     'Offset of destination object name'
  DataLogicalLength  As Long     'Length of bulk data'
  DataLogicalOffset  As Long     'Low offset of bulk data'
  DataLogicalOffset2 As Long     'High offset of bulk data'
End Type
```

StrucId (MQCHAR4) dla MQRMH

Jest to identyfikator struktury struktury nagłówka komunikatu odniesienia. Jest to zawsze pole wejściowe. Jego wartością jest MQRMH_STRUC_ID.

Wartość musi być następująca:

MQRMH_STRUC_ID (ID struktury MQRM)
Identyfikator struktury nagłówka komunikatu odniesienia.

Dla języka programowania C zdefiniowana jest również stała MQRMH_STRUC_ID_ARRAY. Ma taką samą wartość jak MQRMH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQRMH

Numer wersji struktury. Wartość musi być następująca:

MQRMH_VERSION_1
Version-1 -struktura nagłówka komunikatu referencyjnego.

Następująca stała określa numer wersji bieżącej:

MQRMH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka komunikatu referencyjnego.

Wartością początkową tego pola jest MQRMH_VERSION_1.

StrucLength (MQLONG) dla MQRMH

Łączna długość MQRMH, w tym łańcuchy na końcu pól stałych, ale bez danych masowych.

Wartością początkową tego pola jest zero.

Kodowanie (MQLONG) dla MQRMH

Określa kodowanie liczbowe danych masowych. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest MQENC_NATIVE.

CodedCharSetId (MQLONG) dla MQRMH

Określa identyfikator zestawu znaków danych masowych; nie ma zastosowania do danych znakowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dane znakowe w danych następujących po tej strukturze znajdują się w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Nie należy używać pola MQCCSI_INHERIT, jeśli wartością pola `PutApplType` w deskryptorze MQMD jest MQAT_BROKER.

Ta wartość jest obsługiwana w następujących środowiskach:

- AIX
- IBM i
- Linux

- Windows Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQRMH

Określa nazwę formatu danych masowych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w strukturze MQMD.

Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQRMH

Są to flagi komunikatów referencyjnych. Zdefiniowane są następujące opcje:

MQRMHF_LAST
Ta flaga wskazuje, że komunikat odniesienia reprezentuje lub zawiera ostatnią część obiektu odniesienia.

MQRMHF_NOT_LAST (nie)
Komunikat odniesienia nie zawiera ani nie reprezentuje ostatniej części obiektu. Dokumentacja programu MQRMHF_NOT_LAST. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Wartością początkową tego pola jest MQRMHF_NOT_LAST.

ObjectType (MQCHAR8) dla MQRMH

Jest to nazwa, której wyjście komunikatu może użyć do rozpoznania obsługiwanych przez nie typów komunikatów odniesienia. Nazwa musi być zgodna z tymi samymi regułami co pole *Format* (patrz sekcja "Format (MQCHAR8) dla MQRMH" na stronie 571).

Wartością początkową tego pola jest 8 odstępów.

ObjectInstanceId (MQBYTE24) dla MQRMH

To pole służy do identyfikowania konkretnej instancji obiektu. Jeśli nie jest ona potrzebna, należy ją ustawić na następującą wartość:

MQOII_BRAK
Nie określono identyfikatora instancji obiektu. Wartością długości pola jest zero binarne.

Dla języka programowania C zdefiniowana jest również stała MQOII_NONE_ARRAY. Ma ona taką samą wartość jak MQOII_NONE, ale jest tablicą znaków zamiast łańcucha.

Długość tego pola jest określona przez wartość właściwości MQ_OBJECT_INSTANCE_ID_LENGTH. Wartością początkową tego pola jest MQOII_NONE.

SrcEnvDługość (MQLONG) dla MQRMH

Długość danych środowiska źródłowego. Jeśli to pole ma wartość zero, nie ma danych środowiska źródłowego, a parametr *SrcEnvOffset* jest ignorowany.

Wartością początkową tego pola jest 0.

SrcEnvPrzesunięcie (MQLONG)

To pole określa przesunięcie danych środowiska źródłowego od początku struktury MQRMH. Dane środowiska źródłowego mogą być określane przez twórcę komunikatu odniesienia, jeśli są one znane twórcy. Na przykład w systemie Windows dane środowiska źródłowego mogą być ścieżką do katalogu obiektu zawierającego dane masowe. Jeśli jednak twórca nie zna danych środowiska źródłowego,

program obsługi wyjścia komunikatu dostarczony przez użytkownika musi określić wszelkie potrzebne informacje o środowisku.

Długość danych środowiska źródłowego jest określona przez *SrcEnvLength* ; Jeśli ta długość wynosi zero, nie ma danych środowiska źródłowego, a parametr *SrcEnvOffset* jest ignorowany. Jeśli są obecne, dane środowiska źródłowego muszą znajdować się całkowicie w ciągu *StrucLength* bajtów od początku struktury.

Aplikacje nie mogą zakładać, że dane środowiska rozpoczynają się bezpośrednio po ostatnim stałym polu w strukturze lub że sąsiadują z danymi adresowanymi przez pola *SrcNameOffset*, *DestEnvOffset*i *DestNameOffset* .

Wartością początkową tego pola jest 0.

SrcNameDługość (MQLONG) dla MQRMH

Długość nazwy obiektu źródłowego. Jeśli to pole ma wartość zero, nie ma nazwy obiektu źródłowego, a parametr *SrcNameOffset* jest ignorowany.

Wartością początkową tego pola jest 0.

SrcNamePrzesunięcie (MQLONG) dla MQRMH

To pole określa przesunięcie nazwy obiektu źródłowego od początku struktury MQRMH. Nazwa obiektu źródłowego może być określona przez twórcę komunikatu odniesienia, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu źródłowego, wyjście komunikatu dostarczone przez użytkownika musi identyfikować obiekt, do którego ma zostać uzyskany dostęp.

Długość nazwy obiektu źródłowego jest określona przez parametr *SrcNameLength* ; Jeśli ta długość wynosi zero, nie ma nazwy obiektu źródłowego, a parametr *SrcNameOffset* jest ignorowany. Jeśli istnieje, nazwa obiektu źródłowego musi znajdować się w całości w ciągu *StrucLength* bajtów od początku struktury.

Aplikacje nie mogą zakładać, że nazwa obiektu źródłowego jest ciągła w połączeniu z danymi adresowanymi przez pola *SrcEnvOffset*, *DestEnvOffset*i *DestNameOffset* .

Wartością początkową tego pola jest 0.

DestEnvDługość (MQLONG) dla MQRMH

Jest to długość danych środowiska docelowego. Jeśli to pole ma wartość zero, nie ma danych środowiska docelowego, a parametr *DestEnvOffset* jest ignorowany.

Przesunięcie DestEnv(MQLONG) dla MQRMH

To pole określa przesunięcie danych środowiska docelowego od początku struktury MQRMH. Dane środowiska docelowego mogą być określane przez twórcę komunikatu odwołania, jeśli dane te są znane twórcy. Na przykład w systemie Windows dane środowiska docelowego mogą być ścieżką do katalogu obiektu, w którym mają być przechowywane dane masowe. Jeśli jednak twórca nie zna danych środowiska docelowego, to za określenie wymaganych informacji o środowisku odpowiada wyjście komunikatu dostarczone przez użytkownika.

Długość danych środowiska docelowego jest określona przez parametr *DestEnvLength* . Jeśli ta długość wynosi zero, nie ma danych środowiska docelowego, a parametr *DestEnvOffset* jest ignorowany. Jeśli istnieją, dane środowiska docelowego muszą znajdować się całkowicie w obrębie *StrucLength* bajtów od początku struktury.

Aplikacje nie mogą zakładać, że dane środowiska docelowego są ciągłe z danymi adresowanymi przez pola *SrcEnvOffset*, *SrcNameOffset*i *DestNameOffset* .

Wartością początkową tego pola jest 0.

DestNameDługość (MQLONG) dla MQRMH

Długość nazwy obiektu docelowego. Jeśli to pole ma wartość zero, nie ma nazwy obiektu docelowego, a parametr *DestNameOffset* jest ignorowany.

DestNamePrzesunięcie (MQLONG) dla MQRMH

To pole określa przesunięcie nazwy obiektu docelowego od początku struktury MQRMH. Nazwa obiektu docelowego może zostać określona przez twórcę komunikatu odniesienia, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu docelowego, to do wyjścia komunikatu dostarczonego przez użytkownika należy identyfikowanie obiektu, który ma zostać utworzony lub zmodyfikowany.

Długość nazwy obiektu docelowego jest określona przez parametr *DestNameLength* ; Jeśli ta długość wynosi zero, nie ma nazwy obiektu docelowego, a parametr *DestNameOffset* jest ignorowany. Jeśli istnieje, nazwa obiektu docelowego musi znajdować się w całości w ciągu *StrucLength* bajtów od początku struktury.

Aplikacje nie mogą zakładać, że nazwa obiektu docelowego jest ciągła w połączeniu z danymi adresowanymi przez pola *SrcEnvOffset*, *SrcNameOffset*i *DestEnvOffset* .

Wartością początkową tego pola jest 0.

DataLogicalDługość (MQLONG) dla MQRMH

Pole *DataLogicalLength* określa długość danych masowych, do których odwołuje się struktura MQRMH.

Jeśli dane masowe są rzeczywiście obecne w komunikacie, dane rozpoczynają się od przesunięcia *StrucLength* bajtów od początku struktury MQRMH. Długość całego komunikatu minus *StrucLength* określa długość danych masowych.

Jeśli w komunikacie znajdują się dane, parametr *DataLogicalLength* określa ilość istotnych danych. Normalne jest, że *DataLogicalLength* ma taką samą wartość jak długość danych w komunikacie.

Jeśli struktura MQRMH reprezentuje pozostałe dane w obiekcie (począwszy od określonego przesunięcia logicznego), można użyć wartości zero dla parametru *DataLogicalLength*, pod warunkiem, że dane masowe nie są rzeczywiście obecne w komunikacie.

Jeśli nie ma żadnych danych, koniec MQRMH pokrywa się z końcem komunikatu.

Wartością początkową tego pola jest 0.

DataLogical-przesunięcie (MQLONG) dla MQRMH

To pole określa dolne przesunięcie danych masowych od początku obiektu, którego częścią są dane masowe. Przesunięcie danych masowych od początku obiektu jest nazywane *przesunięciem logicznym*. Nie jest to fizyczne przesunięcie danych masowych od początku struktury MQRMH; to przesunięcie jest nadawane przez *StrucLength*.

Aby umożliwić wysyłanie dużych obiektów przy użyciu komunikatów referencyjnych, przesunięcie logiczne jest podzielone na dwa pola, a rzeczywiste przesunięcie logiczne jest określane przez sumę tych dwóch pól:

- *DataLogicalOffset* : reszta uzyskana po podzieleniu przesunięcia logicznego przez 1 000 000 000. Jest to zatem wartość z zakresu od 0 do 999 999 999.
- *DataLogicalOffset2* reprezentuje wynik uzyskany, gdy przesunięcie logiczne zostanie podzielone przez 1 000 000 000. Jest to zatem liczba pełnych wielokrotności 1 000 000 000, które istnieją w przesunięciu logicznym. Liczba wielokrotności mieści się w zakresie od 0 do 999 999 999.

Wartością początkową tego pola jest 0.

DataLogicalOffset2 (MQLONG) dla MQRMH

To pole określa wysokie przesunięcie danych masowych od początku obiektu, którego częścią są dane masowe. Jest to wartość z zakresu od 0 do 999 999 999. Szczegółowe informacje można znaleźć w sekcji *DataLogicalOffset*.

Wartością początkową tego pola jest 0.

MQRR-rekord odpowiedzi

Struktura MQRR służy do odbierania kodu zakończenia i kodu przyczyny będącego wynikiem operacji otwierania lub umieszczania dla pojedynczej kolejki docelowej, gdy miejscem docelowym jest lista dystrybucyjna. MQRR jest strukturą wyjściową dla wywołań MQOPEN, MQPUT i MQPUT1 .

Dostępność

Struktura MQRR jest dostępna na następujących platformach:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

i dla klientów IBM MQ połączonych z tymi systemami.

Zestaw znaków i kodowanie

Dane w MQRR muszą być w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Użycie

Udostępniając tablicę tych struktur w wywołaniach MQOPEN i MQPUT lub w wywołaniu MQPUT1 , można określić kody zakończenia i kody przyczyny dla wszystkich kolejek na liście dystrybucyjnej, gdy wynik wywołania jest mieszany, czyli gdy wywołanie powiedzie się dla niektórych kolejek na liście, ale nie powiedzie się dla innych. Kod przyczyny MQRC_MULTIPLE_REASON z wywołania wskazuje, że rekordy odpowiedzi (jeśli zostały udostępnione przez aplikację) zostały ustawione przez menedżer kolejek.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 524. Pola w MQRR

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
CompCode (kod zakończenia dla kolejki)	MQCC_OK	0
Przyczyna (kod przyczyny kolejki)	MQRC_BRAK	0

Uwagi:

1. W języku programowania C: zmienna makraMQRR_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQRR MyRR = {MQRR_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQRR

```
typedef struct tagMQRR MQRR;
struct tagMQRR {
  MQLONG  CompCode;  /* Completion code for queue */
  MQLONG  Reason;    /* Reason code for queue */
};
```

Deklaracja języka COBOL dla MQRR

```
**   MQRR structure
  10 MQRR.
**    Completion code for queue
   15 MQRR-COMPCODE PIC S9(9) BINARY.
**    Reason code for queue
   15 MQRR-REASON   PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQRR

```
dcl
 1 MQRR based,
  3 CompCode fixed bin(31), /* Completion code for queue */
  3 Reason   fixed bin(31); /* Reason code for queue */
```

Deklaracja Visual Basic dla MQRR

```
Type MQRR
  CompCode As Long 'Completion code for queue'
  Reason   As Long 'Reason code for queue'
End Type
```

CompCode (MQLONG) dla MQRR

Jest to kod zakończenia wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 .

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest MQCC_OK.

Przyczyna (MQLONG) dla MQRR

Jest to kod przyczyny wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR podanej w wywołaniu MQOPEN lub MQPUT1 .

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest MQRC_NONE.

MQSCO-opcje konfiguracyjne SSL/TLS

Struktura MQSCO w połączeniu z polami TLS w strukturze MQCD umożliwia aplikacji działającej jako IBM MQ MQI client określenie opcji konfiguracyjnych, które sterują użyciem protokołu TLS dla połączenia klienta, gdy protokołem kanału jest TCP/IP. Struktura jest parametrem wejściowym wywołania MQCONNX.

Dostępność

Struktura MQSCO jest dostępna dla następujących klientów:

- AIX
- IBM i
- Linux

- Windows

Jeśli protokołem kanału klienta nie jest TCP/IP, struktura MQSCO jest ignorowana.

Zestaw znaków i kodowanie

Dane w obiekcie MQSCO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu **CodedCharSetId** oraz w kodowaniu lokalnego menedżera kolejek określonym przez atrybut MQENC_NATIVE.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 525. Zmienne w MQSCO

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQSCO_STRUC_ID	'SCO¬'
Wersja (numer wersji struktury)	MQSCO_CURRENT_VERSION	1
KeyRepository (położenie repozytorium kluczy)	Brak	Pusty łańcuch lub odstępy
CryptoHardware (szczegóły dotyczące sprzętu szyfrującego)	Brak	Pusty łańcuch lub odstępy
AuthInfoRecCount (liczba rekordów MQAIR)	Brak	0
AuthInfoRecOffset (przesunięcie pierwszego rekordu MQAIR od początku MQSCO)	Brak	0
AuthInfoRecPtr (adres pierwszego rekordu MQAIR)	Brak	Pusty wskaźnik lub puste bajty
Uwaga: Następujące dwa pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_2.		
KeyResetLiczba (liczba operacji resetowania tajnego klucza TLS)	MQSCO_RESET_COUNT_DEFAULT	0
"FipsRequired (MQLONG) dla MQSCO" na stronie 582 (użyj algorytmów szyfrowania z certyfikatem FIPS w produkcie IBM MQ)	MQSSL_FIPS_NO	0
Uwaga: Następujące dwa pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_3.		
EncryptionPolicySuiteB (użyj tylko algorytmów szyfrowania Suite B)	MQ_SUITE_B_NONE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE, MQ_SUITE_B_NOT_AVAILABLE	1, 0, 0, 0

Tabela 525. Zmienne w MQSCO (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
Uwaga: Następujące dwa pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_4.		
StrategiaCertificateVal (strategia sprawdzania poprawności certyfikatu)	MQ_CERT_VAL_POLICY _DEFAULT	0
Uwaga: Następujące dwa pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_5.		
CertificateLabel (szczegóły używanej etykiety certyfikatu)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_6.		
V 9.3.0 V 9.3.0 KeyRepoPasswordPtr (adres hasła repozytorium kluczy TLS)	Brak	Pusty wskaźnik lub puste bajty
V 9.3.0 V 9.3.0 KeyRepoPasswordOffset (przesunięcie hasła repozytorium kluczy TLS)	Brak	0
V 9.3.0 V 9.3.0 KeyRepoPasswordLength (długość hasła repozytorium kluczy TLS)	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQSCO_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQSCO MySCO = {MQSCO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQSCO

```
typedef struct tagMQSCO MQSCO;
struct tagMQSCO {
   MQCHAR4    StrucId;                  /* Structure identifier */
   MQLONG     Version;                  /* Structure version number */
   MQCHAR256  KeyRepository;            /* Location of TLS key */
                                        /* repository */
   MQCHAR256  CryptoHardware;           /* Cryptographic hardware */
                                        /* configuration string */
   MQLONG     AuthInfoRecCount;         /* Number of MQAIR records */
                                        /* present */
   MQLONG     AuthInfoRecOffset;        /* Offset of first MQAIR */
                                        /* record from start of */
                                        /* MQSCO structure */
   PMQAIR     AuthInfoRecPtr;           /* Address of first MQAIR */
                                        /* record */
/* Ver:1 */
   MQLONG     KeyResetCount;            /* Number of unencrypted */
                                        /* bytes sent/received */
                                        /* before secret key is */
                                        /* reset */
   MQLONG     FipsRequired;             /* Using FIPS-certified */
```

```
/* Ver:2 */
                                        /* algorithms */
   MQLONG     EncryptionPolicySuiteB[4];  /* Use only Suite B */
/* Ver:3 */
                                        /* cryptographic algorithms */
   MQLONG     CertificateValPolicy;       /* Certificate validation */
                                        /* policy */
/* Ver:4 */
   MQCHAR64   CertificateLabel;           /* Certificate label */
/* Ver:5 */
   MQPTR      KeyRepoPasswordPtr;         /* Address of key */
                                        /* repository password */
   MQLONG     KeyRepoPasswordOffset;      /* Offset of key repository */
                                        /* password */
   MQLONG     KeyRepoPasswordLength;      /* Length of key repository */
                                        /* password */
/* Ver:6 */
  };
```

Deklaracja języka COBOL dla MQSCO

```
**   MQSCO structure
  10 MQSCO.
**    Structure identifier
   15 MQSCO-STRUCID                 PIC X(4).
**    Structure version number
   15 MQSCO-VERSION                 PIC S9(9) BINARY.
**    Location of TLS key repository
   15 MQSCO-KEYREPOSITORY           PIC X(256).
**    Cryptographic hardware configuration string
   15 MQSCO-CRYPTOHARDWARE          PIC X(256).
**    Number of MQAIR records present
   15 MQSCO-AUTHINFORECCOUNT        PIC S9(9) BINARY.
**    Offset of first MQAIR record from start of MQSCO structure
   15 MQSCO-AUTHINFORECOFFSET       PIC S9(9) BINARY.
**    Address of first MQAIR record
   15 MQSCO-AUTHINFORECPTR          POINTER.
** Version 1 **
**    Number of unencrypted bytes sent/received before secret key is
**    reset
   15 MQSCO-KEYRESETCOUNT           PIC S9(9) BINARY.
**    Using FIPS-certified algorithms
   15 MQSCO-FIPSREQUIRED            PIC S9(9) BINARY.
** Version 2 **
**    Use only Suite B cryptographic algorithms
   15 MQSCO-ENCRYPTIONPOLICYSUITEB  PIC S9(9) BINARY OCCURS 4.
** Version 3 **
**    Certificate validation policy setting
   15 MQSCO-CERTIFICATEVALPOLICY    PIC S9(9) BINARY.
** Version 4 **
**    SSL/TLS certificate label
   15 MQSCO-CERTIFICATELABEL        PIC X(64).
** Version 5 **
**    Add padding to ensure that pointers start on correct
**    boundaries
   15 FILLER                        PIC S9(9) BINARY VALUE 0.
**   Address of key repository password
   15 MQSCO-KEYREPOPASSWORDPTR      POINTER.
**   Offset of key repository password
   15 MQSCO-KEYREPOPASSWORDOFFSET   PIC S9(9) BINARY.
**   Length of key repository password
   15 MQSCO-KEYREPOPASSWORDLENGTH   PIC S9(9) BINARY.
** Version 6 **
```

Deklaracja PL/I dla MQSCO

```
dcl
 1 MQSCO based,
  3 StrucId                   char(4),       /* Structure identifier */
  3 Version                   fixed bin(31), /* Structure version number */
  3 KeyRepository             char(256),     /* Location of TLS key
                                                repository */
  3 CryptoHardware            char(256),     /* Cryptographic hardware
                                                configuration string */
  3 AuthInfoRecCount          fixed bin(31), /* Number of MQAIR records
                                                present */
  3 AuthInfoRecOffset         fixed bin(31), /* Offset of first MQAIR record
```

```
                                                 from start of MQSCO structure */
  3 AuthInfoRecPtr             pointer,       /* Address of first MQAIR record */
  3 KeyResetCount              fixed bin(31), /* Key reset count */
/* Version 1 */
  3 FipsRequired               fixed bin(31), /* FIPS required */
/* Version 2 */
  3 EncryptionPolicySuiteB (4) fixed bin(31), /* Suite B encryption policy */
/* Version 3 */
  3 CertificateValPolicy       fixed bin(31), /* Certificate validation policy */
/* Version 4 */
  3 CertificateLabel           char(64),      /* SSL/TLS certificate label */
/* Version 5 */
  3 KeyRepoPasswordPtr         pointer,       /* Address of key repository
                                                 password */
  3 KeyRepoPasswordOffset      fixed bin(31), /* Offset of key repository
                                                 password */
  3 KeyRepoPasswordLength      fixed bin(31); /* Length of key repository
                                                 password */
/* Version 6 */
```

Deklaracja Visual Basic dla MQSCO

```
Type MQSCO
  StrucId           As String*4   'Structure identifier'
  Version           As Long       'Structure version number'
  KeyRepository     As String*256 'Location of TLS key repository'
  CryptoHardware    As String*256 'Cryptographic hardware configuration'
                                  'string'
  AuthInfoRecCount  As Long       'Number of MQAIR records present'
  AuthInfoRecOffset As Long       'Offset of first MQAIR record from'
                                  'start of MQSCO structure'
  AuthInfoRecPtr    As MQPTR      'Address of first MQAIR record'
  KeyResetCount     As Long       'Number of unencrypted bytes sent/received before secret key
is reset'
'Version 1'
  FipsRequired      As Long       'Mandatory FIPS CipherSpecs?'
'Version 2'
End Type
```

Odsyłacze pokrewne

“MQCNO-opcje połączenia” na stronie 321
Struktura MQCNO umożliwia aplikacji określenie opcji dotyczących połączenia z menedżerem kolejek. Struktura jest parametrem wejścia/wyjścia wywołania MQCONNX.

StrucId (MQCHAR4) dla MQSCO

Jest to identyfikator struktury struktury opcji konfiguracyjnych SSL/TLS. Jest to zawsze pole wejściowe. Jego wartością jest MQSCO_STRUC_ID.

Wartość musi być następująca:

MQSCO_STRUC_ID
Identyfikator struktury opcji konfiguracyjnych SSL/TLS.

Dla języka programowania C zdefiniowana jest również stała MQSCO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQSCO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQSCO

Jest to numer wersji struktury; wartość musi być następująca:

MQSCO_VERSION_1
Version-1 Struktura opcji konfiguracyjnych TLS.

MQSCO_VERSION_2
Version-2 Struktura opcji konfiguracyjnych TLS.

MQSCO_VERSION_3
Version-3 Struktura opcji konfiguracyjnych TLS.

MQSCO_VERSION_4
Version-4 Struktura opcji konfiguracyjnych TLS.

MQSCO_VERSION_5
Version-5 Struktura opcji konfiguracyjnych TLS.

V 9.3.0 V 9.3.0 **MQSCO_VERSION_6**
Version-6 Struktura opcji konfiguracyjnych TLS.

Następująca stała określa numer wersji bieżącej:

MQSCO_CURRENT_VERSION
Bieżąca wersja struktury opcji konfiguracyjnych TLS.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQSCO_VERSION_1.

Multi *KeyRepository (MQCHAR256) dla MQSCO*

To pole dotyczy tylko systemu IBM MQ MQI clients działającego w systemach IBM ii AIX, Linux, and Windows . Określa położenie pliku bazy danych kluczy, w którym przechowywane są klucze i certyfikaty. Jeśli przyrostek pliku nie jest określony, automatycznie dodawany jest przyrostek `.kdb` .

Z każdym plikiem bazy danych kluczy może być skojarzony *plik ukrytych haseł*. Plik ukrytych haseł zawiera zakodowane hasła, które są używane w celu umożliwienia programowego dostępu do bazy danych kluczy. Plik ukrytych haseł musi znajdować się w tym samym katalogu i mieć ten sam rdzeń, co baza danych kluczy, i musi kończyć się przyrostkiem `.sth`.

Jeśli na przykład plik bazy danych kluczy to `/xxx/yyy/key.kdb`, plik ukrytych haseł musi mieć nazwę `/xxx/yyy/key.sth`, gdzie `xxx` i `yyy` reprezentują nazwy katalogów.

V 9.3.0 V 9.3.0 Hasło bazy danych kluczy można również podać w polu *KeyRepoPasswordPtr* lub *KeyRepoPasswordOffset* struktury MQSCO.

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Wartość nie jest sprawdzana. Jeśli wystąpi błąd podczas uzyskiwania dostępu do repozytorium kluczy, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_KEY_REPOSITORY_ERROR.

Aby uruchomić połączenie TLS z serwera IBM MQ MQI client, należy ustawić parametr *KeyRepository* na poprawną nazwę pliku bazy danych kluczy.

Jest to pole wejściowe. Długość tego pola jest określona przez parametr MQ_SSL_KEY_REPOSITORY_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C oraz puste znaki w innych językach programowania.

CryptoHardware (MQCHAR256) dla MQSCO

To pole zawiera szczegóły konfiguracji sprzętu szyfrującego połączonego z systemem klienckim.

Ustaw pole na łańcuch w następującym formacie lub pozostaw je puste lub puste:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting;
```

Aby używać sprzętu szyfrującego, który jest zgodny z interfejsem PKCS #11 , na przykład IBM 4960 lub IBM 4764, należy podać ścieżkę sterownika PKCS #11 , etykietę tokenu PKCS #11 i hasło tokenu PKCS #11 , z których każdy musi być zakończony średnikiem.

Ścieżka sterownika PKCS #11 jest pełną ścieżką do biblioteki współużytkowanej zapewniającej obsługę karty PKCS #11 . Nazwa pliku sterownika PKCS #11 jest nazwą biblioteki współużytkowanej. Przykład wartości wymaganej dla ścieżki PKCS #11 i nazwy pliku:

```
/usr/lib/pkcs11/PKCS11_API.so
```

Etykieta tokenu PKCS #11 musi być zgodna z etykietą, z którą skonfigurowano sprzęt.

Jeśli nie jest wymagana żadna konfiguracja sprzętu szyfrującego, należy ustawić to pole na wartość pustą lub pustą.

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Jeśli wartość jest niepoprawna lub prowadzi do niepowodzenia podczas konfigurowania sprzętu szyfrującego, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_CRYPTO_HARDWARE_ERROR.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_SSL_CRYPTO_HARDWARE_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C oraz puste znaki w innych językach programowania.

AuthInfoRecCount (MQLONG) dla MQSCO

Jest to liczba rekordów informacji uwierzytelniających (MQAIR) zaadresowanych w polach *AuthInfoRecPtr* lub *AuthInfoRecOffset* . Więcej informacji na ten temat zawiera sekcja“MQAIR-rekord informacji uwierzytelniającej” na stronie 274. Wartość musi być równa zero lub większa. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_REC_COUNT_ERROR.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

AuthInfoRecOffset (MQLONG) dla MQSCO

Jest to przesunięcie w bajtach pierwszego rekordu informacji uwierzytelniającej od początku struktury MQSCO. Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli parametr *AuthInfoRecCount* ma wartość zero.

Do określenia rekordów MQAIR można użyć wartości *AuthInfoRecOffset* lub *AuthInfoRecPtr* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola *AuthInfoRecPtr* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

AuthInfoRecPtr (PMQAIR) dla MQSCO

Jest to adres pierwszego rekordu informacji uwierzytelniającej. Pole jest ignorowane, jeśli parametr *AuthInfoRecCount* ma wartość zero.

Tablicę rekordów MQAIR można udostępnić na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *AuthInfoRecPtr*

  W takim przypadku aplikacja może zadeklarować tablicę rekordów MQAIR, która jest oddzielona od struktury MQSCO, i ustawić parametr *AuthInfoRecPtr* na adres tablicy.

  Należy rozważyć użycie języka *AuthInfoRecPtr* dla języków programowania obsługujących typ danych wskaźnika w sposób, który jest przenośny w różnych środowiskach (na przykład w języku programowania C).
- Używając pola przesunięcia *AuthInfoRecOffset*

  W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą obiekt MQSCO, po której następuje tablica rekordów MQAIR, i ustawić parametr *AuthInfoRecOffset* na przesunięcie pierwszego rekordu w tablicy od początku struktury MQSCO. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być ustawiona w MQLONG (najbardziej restrykcyjnym językiem programowania jest COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

  Należy rozważyć użycie języka *AuthInfoRecOffset* dla języków programowania, które nie obsługują typu danych wskaźnika lub implementują typ danych wskaźnika w sposób, który nie jest przenośny w różnych środowiskach (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki można użyć tylko jednej z metod *AuthInfoRecPtr* i *AuthInfoRecOffset* . Wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_AUTH_INFO_REC_ERROR, jeśli obie wartości są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

KeyResetLiczba (MQLONG) dla MQSCO

Reprezentuje łączną liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji TLS przed ponownym negocjowaniem klucza tajnego.

Liczba bajtów obejmuje informacje sterujące wysłane przez agenta MCA.

Jeśli zostanie podana liczba operacji resetowania tajnego klucza TLS z zakresu od 1 bajtu do 32 kB, kanały TLS będą używać liczby operacji resetowania tajnego klucza o wielkości 32 kB. Ma to na celu uniknięcie kosztów przetwarzania nadmiernej liczby operacji resetowania klucza, które miałyby miejsce w przypadku małych wartości resetowania tajnego klucza TLS.

Jest to pole wejściowe. Wartość jest liczbą z zakresu od 0 do 999 999 999, z wartością domyślną 0. Użyj wartości 0, aby wskazać, że klucze tajne nigdy nie są renegocjowane.

FipsRequired (MQLONG) dla MQSCO

Produkt IBM MQ można skonfigurować z użyciem sprzętu szyfrującego w taki sposób, aby używane moduły kryptograficzne były modułami dostarczanymi przez produkt sprzętowy. Moduły te mogą mieć certyfikat FIPS na określonym poziomie w zależności od używanego produktu. To pole służy do określenia, że używane są tylko algorytmy z certyfikatem FIPS, jeśli szyfrowanie jest udostępniane w oprogramowaniu dostarczonym przez IBM MQ.

Uwaga: W systemie AIX, Linux, and Windows IBM MQ zapewnia zgodność ze standardem FIPS 140-2 za pośrednictwem modułu szyfrującego IBM Crypto for C (ICC) . Certyfikat dla tego modułu został przeniesiony do statusu historycznego. Klienci powinni zapoznać się z informacjami w sekcji Certyfikat IBM Crypto for C (ICC) i zapoznać się z poradami NIST. Zastępczy moduł FIPS 140-3 jest obecnie w toku, a jego status można wyświetlić, wyszukując go na liście Moduły NIST CMVP na liście procesów.

Po zainstalowaniu produktu IBM MQ instalowana jest również implementacja szyfrowania TLS, która udostępnia niektóre moduły z certyfikatem FIPS.

Wartości mogą być następujące:

MQSSL_FIPS_NO
Jest to wartość domyślna. W przypadku ustawienia tej wartości:

- Można użyć dowolnej CipherSpec obsługiwanej na konkretnej platformie.
- W przypadku uruchamiania bez użycia sprzętu szyfrującego, CipherSpecs są uruchamiane z użyciem szyfrowania z certyfikatem FIPS 140-2 na platformach IBM MQ .

  Listę certyfikowanych przez FIPS CipherSpecszawiera tabela opisana w sekcji Włączanie specyfikacji szyfrowania CipherSpecs.

MQSSL_FIPS_YES
W przypadku ustawienia tej wartości, o ile do szyfrowania nie jest używany sprzęt szyfrujący, można mieć pewność, że

- W specyfikacji szyfrowania CipherSpec mającej zastosowanie do tego połączenia klienta mogą być używane tylko algorytmy szyfrowania z certyfikatem FIPS.
- Przychodzące i wychodzące połączenia kanału TLS są nawiązywane pomyślnie tylko wtedy, gdy używane są określone specyfikacje szyfru.

  Więcej informacji na ten temat zawiera sekcja Włączanie specyfikacji szyfrowania CipherSpecs .

Uwaga: Jeśli skonfigurowano CipherSpecs tylko dla FIPS, klient MQI odrzuci połączenia, które określają specyfikację szyfrowania CipherSpec z wartością `MQRC_SSL_INITIALIZATION_ERROR`. IBM MQ nie gwarantuje odrzucenia wszystkich takich połączeń i jest odpowiedzialny za określenie, czy konfiguracja IBM MQ jest zgodna ze standardami FIPS.

EncryptionPolicySuiteB (MQLONG) dla MQSCO

To pole określa, czy używane jest szyfrowanie zgodne ze standardem Suite B oraz jaki poziom mocy jest stosowany. Wartość może być jedną lub kilkoma z następujących wartości:

- MQ_SUITE_BNONE

  Szyfrowanie zgodne ze standardem Suite B nie jest używane.
- MQ_SUITE_B_128_BIT

  Używany jest 128-bitowy poziom bezpieczeństwa standardu Suite B.
- MQ_SUITE_B_192_BIT

  Używany jest 192-bitowy poziom bezpieczeństwa standardu Suite B.

Uwaga: Użycie wartości MQ_SUITE_B_NONE z dowolną inną wartością w tym polu jest niepoprawne.

Strategia CertificateVal(MQLONG) dla MQSCO

To pole określa, jaki typ strategii sprawdzania poprawności certyfikatu jest używany.

Pole można ustawić na jedną z następujących wartości:

MQ_CERT_VAL_POLICY_ANY
Zastosuj wszystkie strategie sprawdzania poprawności certyfikatów obsługiwane przez bibliotekę bezpiecznych gniazd. Zaakceptuj łańcuch certyfikatów, jeśli dowolna strategia uzna, że łańcuch certyfikatów jest poprawny.

MQ_CERT_VAL_POLICY_RFC5280
Zastosuj tylko strategię sprawdzania poprawności certyfikatu zgodną ze standardem RFC5280 . To ustawienie zapewnia bardziej rygorystyczne sprawdzanie poprawności niż ustawienie ANY, ale odrzuca niektóre starsze certyfikaty cyfrowe.

Wartością początkową tego pola jest MQ_CERT_VAL_POLICY_ANY

CertificateLabel (MQCHAR64) dla MQSCO

W tym polu znajdują się szczegółowe informacje na temat używanej etykiety certyfikatu.

IBM MQ inicjuje domyślną wartość w polu *CertificateLabel* jako pustą.

Jest on interpretowany w czasie wykonywania jako wartość domyślna i jest kompatybilny wstecz.

Na przykład określenie obiektu MQSCO w wersji wcześniejszej niż 5.0lub użycie domyślnej wartości odstępu w polu *CertificateLabel* powoduje użycie istniejącej wcześniej wartości domyślnej `ibmwebspheremq`*`user_id`*.

V 9.3.0 Multi V 9.3.0 ### *KeyRepoPasswordPtr (MQPTR) dla MQSCO*

Jest to adres w bajtach hasła repozytorium kluczy TLS.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null. To pole jest ignorowane, jeśli wartość *`Version`* jest mniejsza niż MQSCO_VERSION_6.

To pole dotyczy tylko systemu IBM MQ MQI clients działającego w systemach IBM ii AIX, Linux, and Windows .

Frazę hasła repozytorium kluczy można podać w postaci jawnego łańcucha tekstowego lub frazę hasła, która została zaszyfrowana za pomocą programu narzędziowego **`runmqicred`** .

Jeśli zostanie podana zaszyfrowana fraza hasła, należy określić klucz początkowy, który został użyty do zaszyfrowania frazy hasła w strukturze MQCSP udostępnianej przez tę samą aplikację kliencką.

Fraza hasła repozytorium kluczy podana w tym polu nadpisuje wszystkie frazy hasła repozytorium kluczy podane w zmiennej środowiskowej *MQKEYRPWD* lub we właściwości *SSLKeyRepositoryPassword* w sekcji SSL pliku konfiguracyjnego klienta.

Do określenia frazy hasła repozytorium kluczy można użyć wartości *KeyRepoPasswordOffset* lub *KeyRepoPasswordPtr* , ale nie obu tych wartości.

Zadania pokrewne

Podawanie klucza początkowego dla klienta MQI produktu IBM MQ w systemach AIX, Linux i Windows

Ochrona haseł w plikach konfiguracyjnych komponentu IBM MQ

Odsyłacze pokrewne

runmqicred (chroń hasła klienta IBM MQ)

“InitialKeyPtr (MQPTR) dla MQCSP” na stronie 348

Adres początkowego klucza systemu zabezpieczenia hasłem.

V 9.3.0 Multi V 9.3.0 *KeyRepoPasswordOffset (MQLONG) dla MQSCO*

Jest to przesunięcie w bajtach hasła repozytorium kluczy TLS od początku struktury MQSCO. Przesunięcie może być dodatnie lub ujemne.

Do określenia frazy hasła repozytorium kluczy można użyć wartości *KeyRepoPasswordOffset* lub *KeyRepoPasswordPtr* , ale nie obu tych wartości. Więcej informacji na ten temat zawiera opis pola *KeyRepoPasswordPtr* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_6.

V 9.3.0 Multi V 9.3.0 *KeyRepoPasswordLength (MQLONG) dla MQSCO*

Jest to długość frazy hasła repozytorium kluczy TLS.

W systemie IBM imaksymalna długość frazy hasła repozytorium kluczy wynosi 128 znaków. Jeśli fraza hasła repozytorium kluczy jest większa niż maksymalna dozwolona długość, połączenie kończy się niepowodzeniem z błędem MQRC_KEY_REPOSITORY_ERROR.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *Version* jest mniejsza niż MQSCO_VERSION_6.

MQSD-deskryptor subskrypcji

Struktura MQSD służy do określania szczegółów dotyczących budowanej subskrypcji. Struktura jest parametrem wejścia/wyjścia w wywołaniu MQSUB.Więcej informacji na ten temat zawiera sekcja Uwagi dotyczące składni MQSUB.

Dostępność

Struktura MQSD jest dostępna na następujących platformach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows
- z/OS z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Wersja

Bieżąca wersja MQSD to MQSD_VERSION_1.

Zestaw znaków i kodowanie

Dane w usłudze MQSD muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Subskrypcje zarządzane

Jeśli aplikacja nie musi używać określonej kolejki jako miejsca docelowego dla tych publikacji, które są zgodne z jej subskrypcją, może użyć funkcji subskrypcji zarządzanej. Jeśli aplikacja zdecyduje się na użycie subskrypcji zarządzanej, menedżer kolejek poinformuje subskrybenta o miejscu docelowym, do którego są wysyłane opublikowane komunikaty, udostępniając uchwyt obiektu jako dane wyjściowe wywołania MQSUB. Więcej informacji na ten temat zawiera dokument Hobj (MQHOBJ)-input/output.

Po usunięciu subskrypcji menedżer kolejek zobowiązuje się również do czyszczenia komunikatów, które nie zostały pobrane z zarządzanego miejsca docelowego, w następujących sytuacjach:

- Po usunięciu subskrypcji-za pomocą komendy MQCLOSE z opcją MQCO_REMOVE_SUB-i zamknięciu zarządzanego urządzenia Hobj.
- W sposób niejawny oznacza, że w przypadku utraty połączenia z aplikacją korzystającą z subskrypcji nietrwałej (MQSO_NON_DURABLE)
- Po utracie ważności, gdy subskrypcja jest usuwana, ponieważ utraciła ważność, a zarządzany obiekt Hobj jest zamknięty.

Konieczne jest użycie subskrypcji zarządzanych z subskrypcjami nietrwałymi, aby można było wykonać tę procedurę czyszczącą i aby komunikaty dla zamkniętych subskrypcji nietrwałych nie zajmły miejsca w menedżerze kolejek. Trwałe subskrypcje mogą również używać zarządzanych miejsc docelowych.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQSD_STRUC_ID (identyfikator struktury MQS)	'SD¬¬'
Wersja (numer wersji struktury)	MQSD_VERSION_1	1
Opcje (opcje)	MQSO_NON_DURABLE	0
ObjectName (nazwa obiektu)	Brak	Pusty łańcuch lub odstępy
AlternateUserId (alternatywny identyfikator użytkownika)	Brak	Pusty łańcuch lub odstępy
AlternateSecurityId (alternatywny identyfikator zabezpieczeń)	MQSID_BRAK	Wartości null
SubExpiry (utrata ważności subskrypcji)	MQEI_UNLIMITED,	-1
ObjectString (łańcuch obiektu)	Brak	Nazwy i wartości zdefiniowane dla MQCHARV

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
SubName (nazwa subskrypcji)	Brak	Nazwy i wartości zdefiniowane dla MQCHARV
SubUserData (dane użytkownika subskrypcji)	Brak	Nazwy i wartości zdefiniowane dla MQCHARV
SubCorrelId (identyfikator korelacji subskrypcji)	MQCI_BRAK	Wartości null
PubPriority (priorytet publikacji)	MQPRI_PRIORITY_AS_Q_DEF	-3
TokenPubAccounting (token rozliczania publikacji)	MQACT_NONE	Wartości null
PubAppIdentityData (dane tożsamości aplikacji)	Brak	Pusty łańcuch lub odstępy
SelectionString (łańcuch udostępniający kryteria wyboru)	Brak	Nazwy i wartości zdefiniowane dla MQCHARV
SubLevel (poziom subskrypcji)	Brak	1
ResObjectString (długa nazwa obiektu)	Brak	Nazwy i wartości zdefiniowane dla MQCHARV

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQSD_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQSD MySD = {MQSD_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQSD

```
typedef struct tagMQSD MQSD;
struct tagMQSD {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    Options;              /* Options associated with subscribing */
  MQCHAR48  ObjectName;           /* Object name */
  MQCHAR12  AlternateUserId;      /* Alternate user identifier */
  MQBYTE40  AlternateSecurityId;  /* Alternate security identifier */
  MQLONG    SubExpiry;            /* Expiry of Subscription */
  MQCHARV   ObjectString;         /* Object Long name */
  MQCHARV   SubName;              /* Subscription name */
  MQCHARV   SubUserData;          /* Subscription User data */
  MQBYTE24  SubCorrelId;          /* Correlation Id related to this subscription */
  MQLONG    PubPriority;          /* Priority set in publications */
  MQBYTE32  PubAccountingToken;   /* Accounting Token set in publications */
  MQCHAR32  PubApplIdentityData;  /* Appl Identity Data set in publications */
  MQCHARV   SelectionString;      /* Message selector structure */
  MQLONG    SubLevel;             /* Subscription level */
  MQCHARV   ResObjectString;      /* Resolved Long object name*/
```

```
  /* Ver:1 */
};
```

Deklaracja języka COBOL dla MQSD

```
** Address of variable length string
   20  MQSD-OBJECTSTRING-VSPTR         POINTER.
** Offset of variable length string
   20  MQSD-OBJECTSTRING-VSOFFSET      PIC S9(9) BINARY.
** size of buffer
   20  MQSD-OBJECTSTRING-VSBUFSIZE     PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-OBJECTSTRING-VSLENGTH      PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-OBJECTSTRING-VSCCSID       PIC S9(9) BINARY.
** Subscription name
 15  MQSD-SUBNAME.
** Address of variable length string
   20  MQSD-SUBNAME-VSPTR              POINTER.
** Offset of variable length string
   20  MQSD-SUBNAME-VSOFFSET           PIC S9(9) BINARY.
** size of buffer
   20  MQSD-SUBNAME-VSBUFSIZE          PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-SUBNAME-VSLENGTH           PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-SUBNAME-VSCCSID            PIC S9(9) BINARY.
** Subscription User data
 15  MQSD-SUBUSERDATA.
** Address of variable length string
   20  MQSD-SUBUSERDATA-VSPTR          POINTER.
** Offset of variable length string
   20  MQSD-SUBUSERDATA-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
   20  MQSD-SUBUSERDATA-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-SUBUSERDATA-VSLENGTH       PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-SUBUSERDATA-VSCCSID        PIC S9(9) BINARY.
** Correlation Id related to this subscription
 15  MQSD-SUBCORRELID                 PIC X(24).
** Priority set in publications
 15  MQSD-PUBPRIORITY                 PIC S9(9) BINARY.
** Accounting Token set in publications
 15  MQSD-PUBACCOUNTINGTOKEN          PIC X(32).
** Appl Identity Data set in publications
 15  MQSD-PUBAPPLIDENTITYDATA         PIC X(32).
** Message Selector
 15 MQSD-SELECTIONSTRING.
** Address of variable length string
   20  MQSD-SELECTIONSTRING-VSPTR     POINTER.
** Offset of variable length string
   20  MQSD-SELECTIONSTRING-VSOFFSET       PIC S9(9) BINARY.
** size of buffer
   20  MQSD-SELECTIONSTRING-VSBUFSIZE      PIC S9(9) BINARY.
** Length of variable length string
   20  MQSD-SELECTIONSTRING-VSLENGTH       PIC S9(9) BINARY.
** CCSID of variable length string
   20  MQSD-SELECTIONSTRING-VSCCSID        PIC S9(9) BINARY.
** Selection criteria
   20  MQSD-SELECTIONSTRING-SUBLEVEL       PIC S9(9) BINARY.
** Long object name
   20  MQSD-SELECTIONSTRING-RESOBJSTRING   PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla usługi MQSD

```
dcl
1 MQSD based,
3 StrucId       char(4),    /* Structure identifier */
3 Version       fixed bin(31), /* Structure version number */
3 Options       fixed bin(31), /* Options associated with subscribing */
3 ObjectName     char(48),   /* Object name */
3 AlternateUserId   char(12),   /* Alternate user identifier */
3 AlternateSecurityId char(40),   /* Alternate security identifier */
3 SubExpiry      fixed bin(31), /* Expiry of Subscription */
3 ObjectString,           /* Object Long name */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
```

```
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */
3 SubName,              /* Subscription name */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */
3 SubUserData,            /* Subscription User data */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31), /* CCSID of variable length string */
3 SubCorrelId     char(24),   /* Correlation Id related to this subscription */
3 PubPriority     fixed bin(31), /* Priority set in publications */
3 PubAccountingToken char(32),   /* Accounting Token set in publications */
3 PubApplIdentityData char(32),   /* Appl Identity Data set in publications */
3 SelectionString,          /* Message Selection */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31), /* CCSID of variable length string */
3 SubLevel      fixed bin(31), /* Subscription level */
3 ResObjectString,          /* Resolved Long object name */
5 VSPtr       pointer,    /* Address of variable length string */
5 VSOffset      fixed bin(31), /* Offset of variable length string */
5 VSBufSize     fixed bin(31), /* size of buffer */
5 VSLength      fixed bin(31), /* Length of variable length string */
5 VSCCSID      fixed bin(31); /* CCSID of variable length string */
```

Deklaracja High Level Assembler dla MQSD

```
MQSD              DSECT
MQSD_STRUCID          DS  CL4  Structure identifier
MQSD_VERSION          DS  F   Structure version number
MQSD-OPTIONS          DS  F   Options associated with subscribing
MQSD_OBJECTNAME        DS  CL48 Object name
MQSD_ALTERNATEUSERID      DS  CL12 Alternate user identifier
MQSD_ALTERNATESECURITYID    DS  CL40 Alternate security identifier
MQSD_SUBEXPIRY         DS  F   Expiry of Subscription
MQSD_OBJECTSTRING       DS  0F  Object Long name
MQSD_OBJECTSTRING_VSPTR    DS  F   Address of variable length string
MQSD_OBJECTSTRING_VSOFFSET   DS  F   Offset of variable length string
MQSD_OBJECTSTRING_VSBUFSIZE  DS  F   size of buffer
MQSD_OBJECTSTRING_VSLENGTH   DS  F   Length of variable length string
MQSD_OBJECTSTRING_VSCCSID   DS  F   CCSID of variable length string
MQSD_OBJECTSTRING_LENGTH    EQU  *-MQSD_OBJECTSTRING
ORG  MQSD_OBJECTSTRING
MQSD_OBJECTSTRING_AREA     DS  CL(MQSD_OBJECTSTRING_LENGTH)
*
MQSD_SUBNAME          DS  0F  Subscription name
MQSD_SUBNAME_VSPTR       DS  F   Address of variable length string
MQSD_SUBNAME_VSOFFSET     DS  F   Offset of variable length string
MQSD_SUBNAME_VSBUFSIZE     DS  F   size of buffer
MQSD_SUBNAME_VSLENGTH     DS  F   Length of variable length string
MQSD_SUBNAME_VSCCSID      DS  F   CCSID of variable length string
MQSD_SUBNAME_LENGTH      EQU  *-MQSD_SUBNAME
ORG  MQSD_SUBNAME
MQSD_SUBNAME_AREA       DS  CL(MQSD_SUBNAME_LENGTH)
*
MQSD_SUBUSERDATA        DS  0F  Subscription User data
MQSD_SUBUSERDATA_VSPTR     DS  F   Address of variable length string
MQSD_SUBUSERDATA_VSOFFSET   DS  F   Offset of variable length string
MQSD_SUBUSERDATA_VSBUFSIZE   DS  F   size of buffer
MQSD_SUBUSERDATA_VSLENGTH   DS  F   Length of variable length string
MQSD_SUBUSERDATA_VSCCSID    DS  F   CCSID of variable length string
MQSD_SUBUSERDATA_LENGTH    EQU  *-MQSD_SUBUSERDATA
ORG  MQSD_SUBUSERDATA
MQSD_SUBUSERDATA_AREA     DS  CL(MQSD_SUBUSERDATA_LENGTH)
*
MQSD_SUBCORRELID        DS  CL24 Correlation Id related to this subscription
MQSD_PUBPRIORITY        DS  F   Priority set in publications
MQSD_PUBACCOUNTINGTOKEN    DS  CL32 Accounting Token set in publications
MQSD_PUBAPPLIDENTITYDATA    DS  CL32 Appl Identity Data set in publications
*
MQSD_SELECTIONSTRING      DS  F   Message Selector
MQSD_SELECTIONSTRING_VSPTR   DS  F   Address of variable length string
```

```
MQSD_SELECTIONSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_SELECTIONSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_SELECTIONSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_SELECTIONSTRING_VSCCSID  DS  F   CCSID of variable length string
MQSD_SELECTIONSTRING_LENGTH  EQU *- MQSD_SELECTIONSTRING
ORG MQSD_SELECTIONSTRING
MQSD_SELECTIONSTRING_AREA   DS  CL(MQSD_SELECTIONSTRING_LENGTH)
*
MQSD-SUBLEVEL         DS  F   Subscription level
*
MQSD_RESOBJECTSTRING      DS  F   Resolved Long object name
MQSD_RESOBJECTSTRING_VSPTR   DS  F   Address of variable length string
MQSD_RESOBJECTSTRING_VSOFFSET DS  F   Offset of variable length string
MQSD_RESOBJECTSTRING_VSBUFSIZE DS  F   size of buffer
MQSD_RESOBJECTSTRING_VSLENGTH DS  F   Length of variable length string
MQSD_RESOBJECTSTRING_VSCCSID  DS  F   CCSID of variable length string
MQSD_RESOBJECTSTRING_LENGTH  EQU *- MQSD_RESOBJECTSTRING
ORG MQSD_RESOBJECTSTRING
MQSD_RESOBJECTSTRING_AREA   DS  CL(MQSD_RESOBJECTSTRING_LENGTH)
*
MQSD_LENGTH          EQU  *-MQSD
ORG  MQSD
MQSD_AREA           DS  CL(MQSD_LENGTH)
```

StrucId (MQCHAR4) dla MQSD

Jest to identyfikator struktury struktury deskryptora subskrypcji. Jest to zawsze pole wejściowe. Jego wartością jest MQSD_STRUC_ID.

Wartość musi być następująca:

MQSD_STRUC_ID (identyfikator struktury MQS)
Identyfikator struktury deskryptora subskrypcji.

Dla języka programowania C zdefiniowana jest również stała MQSD_STRUC_ID_ARRAY. Ma taką samą wartość jak MQSD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQSD

Jest to numer wersji struktury; wartość musi być następująca:

MQSD_VERSION_1
Version-1 Struktura deskryptora subskrypcji.

Następująca stała określa numer wersji bieżącej:

MQSD_CURRENT_VERSION
Bieżąca wersja struktury deskryptora subskrypcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQSD_VERSION_1.

Opcje (MQLONG) dla MQSD

Ta opcja udostępnia opcje umożliwiające sterowanie działaniem wywołania MQSUB.

Należy podać co najmniej jedną z następujących opcji:

- MQSO_ALTER (zmiana MQSO)
- MQSO_RESUME
- MQSO_CREATE

Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

Kombinacje, które nie są poprawne, zostały opisane w tym temacie; wszystkie inne kombinacje są poprawne.

Opcje dostępu lub tworzenia: Opcje dostępu i tworzenia określają, czy subskrypcja jest tworzona, czy też istniejąca subskrypcja jest zwracana lub zmieniana. Należy określić co najmniej jedną z tych opcji.

Tabela 526. Poprawne kombinacje opcji dostępu i tworzenia	
Kombinacja opcji	**Uwagi**
MQSO_CREATE	Tworzy subskrypcję, jeśli nie istnieje. Ta kombinacja nie powiedzie się, jeśli subskrypcja istnieje.
MQSO_RESUME	Wznawia istniejącą subskrypcję. Ta kombinacja nie powiedzie się, jeśli nie istnieje żadna subskrypcja.
MQSO_CREATE + MQSO_RESUME	Tworzy subskrypcję, jeśli taka subskrypcja nie istnieje, i wznawia ją, jeśli taka subskrypcja istnieje. Ta kombinacja jest przydatna, gdy jest używana w aplikacji, która jest uruchamiana wiele razy.
MQSO_ALTER (patrz uwaga)	Wznawia istniejącą subskrypcję, zmieniając wszystkie pola w taki sposób, aby były zgodne z określonymi w pliku MQSD. Ta kombinacja nie powiedzie się, jeśli nie istnieje żadna subskrypcja.
MQSO_CREATE + MQSO_ALTER (patrz uwaga)	Tworzy subskrypcję, jeśli taka subskrypcja nie istnieje, i wznawia ją (jeśli taka subskrypcja istnieje), zmieniając pola w taki sposób, aby były zgodne z określonymi w pliku MQSD. Ta kombinacja jest przydatna, gdy jest używana w aplikacji, która chce mieć pewność, że jej subskrypcja jest w określonym stanie przed kontynuowaniem.

Uwaga:

Opcje określające opcję MQSO_ALTER mogą również określać opcję MQSO_RESUME, ale ta kombinacja nie ma dodatkowego wpływu na określanie samej opcji MQSO_ALTER. Instrukcja MQSO_ALTER implikuje opcję MQSO_RESUME, ponieważ wywołanie komendy MQSUB w celu zmiany subskrypcji implikuje, że subskrypcja również zostanie wznowiona. Sytuacja odwrotna nie jest prawdziwa, jednak wznowienie subskrypcji nie oznacza, że należy ją zmienić.

MQSO_CREATE

Utwórz nową subskrypcję dla określonego tematu. Jeśli istnieje subskrypcja używająca tej samej zmiennej *SubName* , wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_SUB_ALREADY_EXISTS. Tego niepowodzenia można uniknąć, łącząc opcję MQSO_CREATE z opcją MQSO_RESUME. *SubName* nie zawsze jest konieczne. Więcej informacji na ten temat zawiera opis tego pola.

Połączenie operacji MQSO_CREATE z opcją MQSO_RESUME zwraca uchwyt do istniejącej subskrypcji dla określonego elementu *SubName* , jeśli taka subskrypcja zostanie znaleziona. Jeśli nie istnieje, zostanie utworzona nowa subskrypcja przy użyciu wszystkich pól udostępnionych w pliku MQSD.

Podobnie można połączyć instrukcję MQSO_CREATE z instrukcją MQSO_ALTER.

MQSO_RESUME

Zwraca uchwyt do istniejącej subskrypcji, która jest zgodna z subskrypcją określoną przez parametr *SubName*. W zgodnych atrybutach subskrypcji nie są wprowadzane żadne zmiany i są one zwracane w danych wyjściowych w strukturze MQSD. Używane są tylko następujące pola MQSD: StrucId, Version, Options, AlternateUserId i AlternateSecurityId oraz SubName.

Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_NO_SUBSCRIPTION, jeśli subskrypcja nie istnieje i nie jest zgodna z pełną nazwą subskrypcji. Tego niepowodzenia można uniknąć, łącząc opcję MQSO_CREATE z opcją MQSO_RESUME.

Identyfikator użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika ostatniej pomyślnej zmiany. Jeśli używany jest identyfikator AlternateUseri dla tego

użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkowników, alternatywny identyfikator użytkownika jest rejestrowany jako identyfikator użytkownika, który utworzył subskrypcję, a nie jako identyfikator użytkownika, dla którego została utworzona subskrypcja.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji MQSO_ANY_USERID, a identyfikator użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu subskrypcji, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_IDENTITY_MISMATCH.

Jeśli zgodna subskrypcja istnieje i jest obecnie używana, wywołanie nie powiedzie się i zostanie zwrócony komunikat MQRC_SUBSCRIPTION_IN_USE.

Jeśli subskrypcja o nazwie określonej w polu SubName nie jest poprawną subskrypcją, która może zostać wznowiona lub zmieniona w aplikacji, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_INVALID_SUBSCRIPTION.

Opcja MQSO_RESUME jest implikowana przez instrukcję MQSO_ALTER, dlatego nie trzeba jej łączyć z tą opcją. Jednak połączenie tych dwóch opcji nie powoduje błędu.

MQSO_ALTER (zmiana MQSO)

Zwraca uchwyt do istniejącej subskrypcji z pełną nazwą subskrypcji zgodną z nazwą określoną w parametrze *SubName*. Wszystkie atrybuty subskrypcji, które są inne niż określone w pliku MQSD, są zmieniane w subskrypcji, chyba że zmiana jest niedozwolona dla tego atrybutu. Szczegóły znajdują się w opisie każdego atrybutu i są podsumowane w poniższej tabeli. Próba zmiany atrybutu, którego nie można zmienić, lub zmiany subskrypcji, dla której ustawiono opcję MQSO_IMMUTABLE, zakończy się niepowodzeniem z kodem przyczyny przedstawionym w poniższej tabeli.

Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_NO_SUBSCRIPTION, jeśli subskrypcja zgodna z pełną nazwą subskrypcji nie istnieje. Niepowodzenie to można uniknąć, łącząc opcję MQSO_CREATE z opcją MQSO_ALTER.

Połączenie MQSO_CREATE z MQSO_ALTER zwraca uchwyt do istniejącej subskrypcji dla określonego elementu *SubName* (jeśli zostanie znaleziony). Jeśli nie istnieje żadna subskrypcja, zostanie utworzona nowa subskrypcja przy użyciu wszystkich pól udostępnionych w obiekcie MQSD.

Identyfikator użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub, jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika ostatniej pomyślnej zmiany. Jeśli używany jest identyfikator AlternateUseri dla tego użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkowników, to alternatywny identyfikator użytkownika jest rejestrowany jako identyfikator użytkownika, który utworzył subskrypcję, a nie jako identyfikator użytkownika, który utworzył subskrypcję.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji MQSO_ANY_USERID, a identyfikator użytkownika subskrypcji jest inny niż identyfikator użytkownika aplikacji żądającej uchwytu do subskrypcji, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_IDENTITY_MISMATCH.

Jeśli zgodna subskrypcja istnieje i jest obecnie używana, wywołanie nie powiedzie się i zostanie zwrócony komunikat MQRC_SUBSCRIPTION_IN_USE.

Jeśli subskrypcja o nazwie określonej w polu SubName nie jest poprawną subskrypcją, która może zostać wznowiona lub zmieniona w aplikacji, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_INVALID_SUBSCRIPTION.

W poniższej tabeli przedstawiono możliwość zmiany wartości atrybutów MQSO_ALTER w MQSD i MQSUB.

Tabela 527. Atrybuty w tabelach MQSD i MQSUB, które można zmieniać

Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Ten atrybut można zmienić za pomocą komendy MQSO_ALTER	Kod przyczyny
Usługa MQSD	Opcje trwałości	Nie	MQRC_DURABILITY_NOT_ALTERABLE
Usługa MQSD	Opcje miejsca docelowego	Tak	Brak

Tabela 527. Atrybuty w tabelach MQSD i MQSUB, które można zmieniać (kontynuacja)

Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Ten atrybut można zmienić za pomocą komendy MQSO_ALTER	Kod przyczyny
Usługa MQSD	Opcje rejestracji	Tak (patrz uwaga "1" na stronie 592)	MQRC_GROUPING_NOT_ALTERABLE, jeśli próbujesz zmienić MQSO_GROUP_SUB
Usługa MQSD	Opcje publikacji	Tak (patrz uwaga "2" na stronie 592)	Brak
Usługa MQSD	Opcje znaku wieloznacznego	Nie	MQRC_TOPIC_NOT_ALTERABLE (nie MOŻNA)
Usługa MQSD	Inne opcje	Nie (patrz uwaga "3" na stronie 592)	Brak
Usługa MQSD	ObjectName	Nie	MQRC_TOPIC_NOT_ALTERABLE (nie MOŻNA)
Usługa MQSD	Identyfikator AlternateUser	Nie (patrz uwaga "4" na stronie 592)	Brak
Usługa MQSD	Identyfikator AlternateSecurity	Nie (patrz uwaga "4" na stronie 592)	Brak
Usługa MQSD	SubExpiry	Tak	Brak
Usługa MQSD	ObjectString	Nie	MQRC_TOPIC_NOT_ALTERABLE (nie MOŻNA)
Usługa MQSD	SubName	Nie (patrz uwaga "5" na stronie 592)	Brak
Usługa MQSD	Dane SubUser	Tak	Brak
Usługa MQSD	SubCorrelIdentyfikator	Tak (patrz uwaga "6" na stronie 592)	MQRC_GROUPING_NOT_ALTERABLE, gdy w zgrupowanej subskrypcji
Usługa MQSD	PubPriority	Tak	Brak
Usługa MQSD	Token PubAccounting	Tak	Brak
Usługa MQSD	PubApplIdentityData	Tak	Brak
Usługa MQSD	SubLevel	Nie	MQRC_SUBLEVEL_NOT_ALTERABLE (nie MOŻNA)
MQSUB	Hobj	Tak (patrz uwaga "6" na stronie 592)	MQRC_GROUPING_NOT_ALTERABLE, gdy w zgrupowanej subskrypcji

Uwagi:

1. Nie można zmienić tabeli MQSO_GROUP_SUB.
2. Nie można zmienić elementu MQSO_NEW_PUBLIC ONLY, ponieważ nie jest on częścią subskrypcji
3. Te opcje nie są częścią subskrypcji
4. Ten atrybut nie jest częścią subskrypcji
5. Ten atrybut jest tożsamością zmienianej subskrypcji
6. Możliwość zmiany, z wyjątkiem sytuacji, gdy część zgrupowanego elementu podrzędnego (MQSO_GROUP_SUB)

Opcje trwałości: Następujące opcje sterują trwałością subskrypcji. Można określić tylko jedną z tych opcji. Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji MQSO_ALTER, nie można zmienić trwałości subskrypcji. Po powrocie z wywołania MQSUB używającego komendy MQSO_RESUME ustawiana jest odpowiednia opcja trwałości.

MQSO_DURABLE
Zażądaj, aby subskrypcja tego tematu pozostała, dopóki nie zostanie jawnie usunięta za pomocą komendy MQCLOSE z opcją MQCO_REMOVE_SUB. Jeśli ta subskrypcja nie zostanie jawnie usunięta, pozostanie ona nawet po zamknięciu połączenia aplikacji z menedżerem kolejek.

Jeśli do tematu, który jest zdefiniowany jako nie zezwalający na trwałe subskrypcje, zażądano trwałej subskrypcji, wywołanie kończy się niepowodzeniem z błędem MQRC_DURABILITY_NOT_ALLOWED.

MQSO_NON_DURABLE
Zażądaj, aby subskrypcja tego tematu została usunięta podczas zamykania połączenia aplikacji z menedżerem kolejek, jeśli nie została jeszcze jawnie usunięta. MQSO_NON_DURABLE jest

przeciwieństwem opcji MQSO_DURABLE i jest definiowany w celu wspomagania dokumentacji programu. Jest to wartość domyślna, jeśli nie określono żadnej z nich.

Opcje miejsca docelowego: Poniższa opcja steruje miejscem docelowym, do którego są wysyłane publikacje dla tematu, który został zasubskrybowany. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER można zmienić miejsce docelowe używane na potrzeby publikacji dla subskrypcji. W przypadku powrotu z wywołania MQSUB używającego opcji MQSO_RESUME ta opcja jest ustawiana w razie potrzeby.

MQSO_MANAGED

Zażądaj, aby miejsce docelowe, do którego są wysyłane publikacje, było zarządzane przez menedżer kolejek.

Uchwyt obiektu zwracany w programie *Hobj* reprezentuje kolejkę zarządzaną przez menedżer kolejek i jest używany z kolejnymi wywołaniami MQGET, MQCB, MQINQ lub MQCLOSE.

Nie można podać uchwytu obiektu zwróconego z poprzedniego wywołania MQSUB w parametrze **Hobj** , jeśli nie określono parametru MQSO_MANAGED.

MQSO_NO_MULTICAST

Żądanie, aby miejsce docelowe, do którego są wysyłane publikacje, nie było adresem grupy rozsyłania grupowego. Ta opcja jest poprawna tylko w połączeniu z opcją MQSO_MANAGED. Jeśli w parametrze **Hobj** podano uchwyt kolejki, nie można użyć rozsyłania grupowego dla tej subskrypcji, a opcja nie jest poprawna.

Jeśli temat jest zdefiniowany w taki sposób, aby zezwalał tylko na subskrypcje rozsyłania grupowego przy użyciu ustawienia `MCAST(ONLY)` , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MULTICAST_REQUIRED.

Opcja zasięgu: Poniższa opcja steruje zasięgiem dokonanej subskrypcji. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER nie można zmienić tej opcji zasięgu subskrypcji. Po powrocie z wywołania MQSUB przy użyciu opcji MQSO-RESUME ustawiana jest odpowiednia opcja zasięgu.

MQSO_SCOPE_QMGR,

Ta subskrypcja jest dokonywana tylko w lokalnym menedżerze kolejek. Żadna subskrypcja proxy nie jest dystrybuowana do innych menedżerów kolejek w sieci. Do tego subskrybenta wysyłane są tylko publikacje opublikowane w tym menedżerze kolejek. Powoduje to przesłonięcie dowolnego zachowania ustawionego przy użyciu atrybutu tematu SUBSCOPE.

Uwaga: Jeśli nie jest ustawiony, zasięg subskrypcji jest określany przez atrybut tematu SUBSCOPE.

Opcje rejestracji: Następujące opcje sterują szczegółami rejestracji dokonanej w menedżerze kolejek dla tej subskrypcji. W przypadku zmiany istniejącej subskrypcji przy użyciu opcji MQSO_ALTER można zmienić te opcje rejestracji. Po powrocie z wywołania MQSUB używającego komendy MQSO_RESUME ustawione są odpowiednie opcje rejestracji.

MQSO_GROUP_SUB

Ta subskrypcja ma być zgrupowana z innymi subskrypcjami tego samego SubLevel przy użyciu tej samej kolejki i tego samego identyfikatora korelacji, tak aby wszystkie publikacje tematów, które spowodowałyby dostarczenie więcej niż jednego komunikatu publikacji do grupy subskrypcji, ze względu na nakładający się zestaw używanych łańcuchów tematów, powodowały dostarczenie do kolejki tylko jednego komunikatu. Jeśli ta opcja nie jest używana, każda unikalna subskrypcja (identyfikowana przez SubName), która jest zgodna, jest dostarczana z kopią publikacji, co może oznaczać, że więcej niż jedna kopia publikacji może zostać umieszczona w kolejce współużytkowanej przez pewną liczbę subskrypcji.

Tylko najbardziej znacząca subskrypcja w grupie jest dostarczana wraz z kopią publikacji. Najbardziej znacząca subskrypcja jest oparta na pełnej nazwie tematu aż do miejsca, w którym znaleziono znak wieloznaczny. Jeśli w grupie używane są różne schematy znaków wieloznacznych, ważna jest tylko pozycja znaku wieloznacznego. Zaleca się, aby nie łączyć różnych schematów znaków wieloznacznych w obrębie grupy subskrypcji, które współużytkują tę samą kolejkę.

Podczas tworzenia nowej zgrupowanej subskrypcji musi ona nadal mieć unikalną nazwę SubName, ale jeśli jest zgodna z pełną nazwą tematu istniejącej subskrypcji w grupie, wywołanie kończy się niepowodzeniem z wartością MQRC_DUPLICATE_GROUP_SUB.

Jeśli w najbardziej znaczącej subskrypcji w grupie określono również wartość MQSO_NOT_OWN_PUBS i jest to publikacja z tej samej aplikacji, do kolejki nie jest dostarczana żadna publikacja.

Podczas modyfikowania subskrypcji dokonanej za pomocą tej opcji nie można zmieniać pól, które implikują grupowanie, hobby w wywołaniu MQSUB (reprezentujących nazwę kolejki i menedżera kolejek) oraz identyfikatora SubCorrel. Próba ich zmiany spowoduje, że wywołanie nie powiedzie się i zostanie nawiązane połączenie z MQRC_GROUPING_NOT_ALTERABLE.

Ta opcja musi być połączona z opcją MQSO_SET_CORREL_ID z identyfikatorem SubCorrel, który nie jest ustawiony na wartość MQCI_NONE i nie może być łączony z opcją MQSO_MANAGED.

MQSO_ANY_USERID

Jeśli określono opcję MQSO_ANY_USERID, tożsamość subskrybenta nie jest ograniczona do pojedynczego identyfikatora użytkownika. Dzięki temu każdy użytkownik może zmienić lub wznowić subskrypcję, gdy ma odpowiednie uprawnienia. Subskrypcja może być dostępna tylko dla jednego użytkownika w danym momencie. Próba wznowienia korzystania z subskrypcji aktualnie używanej przez inną aplikację powoduje, że wywołanie nie powiedzie się i zostanie użyta komenda MQRC_SUBSCRIPTION_IN_USE.

Aby dodać tę opcję do istniejącej subskrypcji, wywołanie MQSUB (za pomocą MQSO_ALTER) musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli wywołanie MQSUB odwołuje się do istniejącej subskrypcji z ustawioną wartością MQSO_ANY_USERID, a identyfikator użytkownika różni się od oryginalnej subskrypcji, wywołanie powiedzie się tylko wtedy, gdy nowy identyfikator użytkownika ma uprawnienie do subskrybowania tematu. Po pomyślnym zakończeniu działania przyszłe publikacje dla tego subskrybenta są umieszczane w kolejce subskrybentów z nowym identyfikatorem użytkownika ustawionym w komunikacie o publikacji.

Nie należy określać jednocześnie identyfikatorów MQSO_ANY_USERID i MQSO_FIXED_USERID. Jeśli żadna z tych wartości nie zostanie podana, wartością domyślną jest MQSO_FIXED_USERID.

MQSO_FIXED_USERID (Identyfikator stałego użytkownika)

Jeśli określono parametr MQSO_FIXED_USERID, subskrypcja może zostać zmieniona lub wznowiona tylko przez ostatniego użytkownika, który ją zmodyfikuje. Jeśli subskrypcja nie została zmieniona, jest to identyfikator użytkownika, który ją utworzył.

Jeśli komenda MQSUB odwołuje się do istniejącej subskrypcji z ustawioną wartością MQSO_ANY_USERID i modyfikuje subskrypcję za pomocą komendy MQSO_ALTER w celu użycia opcji MQSO_FIXED_USERID, identyfikator użytkownika subskrypcji jest teraz poprawiany przy użyciu tego nowego identyfikatora użytkownika. Wywołanie powiedzie się tylko wtedy, gdy nowy ID użytkownika ma uprawnienia do subskrybowania tematu.

Jeśli ID użytkownika inny niż zarejestrowany jako właściciel subskrypcji próbuje wznowić lub zmienić subskrypcję MQSO_FIXED_USERID, wywołanie kończy się niepowodzeniem z błędem MQRC_IDENTITY_MISMATCH. Identyfikator użytkownika będącego właścicielem subskrypcji można wyświetlić za pomocą komendy DISPLAY SBSTATUS.

Nie należy określać jednocześnie identyfikatorów MQSO_ANY_USERID i MQSO_FIXED_USERID. Jeśli żadna z tych wartości nie zostanie podana, wartością domyślną jest MQSO_FIXED_USERID.

Opcje publikacji: Następujące opcje sterują sposobem wysyłania publikacji do tego subskrybenta. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER można zmienić te opcje publikowania.

MQSO_NOT_OWN_PUBS:

Informuje broker, że aplikacja nie chce wyświetlać żadnych własnych publikacji. Publikacje są uznawane za pochodzące z tej samej aplikacji, jeśli uchwyty połączenia są takie same. W przypadku

powrotu z wywołania MQSUB używającego opcji MQSO_RESUME ta opcja jest ustawiana w razie potrzeby.

MQSO_NEW_PUBLIC TYLKO
Podczas tworzenia tej subskrypcji nie są wysyłane żadne zachowane publikacje, a tylko nowe. Ta opcja ma zastosowanie tylko wtedy, gdy określono opcję MQSO_CREATE. Wszelkie kolejne zmiany w subskrypcji nie wpływają na przepływ publikacji, dlatego wszystkie publikacje przechowywane w temacie zostaną już wysłane do subskrybenta jako nowe publikacje.

Jeśli ta opcja zostanie podana bez opcji MQSO_CREATE, wywołanie zakończy się niepowodzeniem z błędem MQRC_OPTIONS_ERROR. Po powrocie z wywołania MQSUB używającego opcji MQSO_RESUME ta opcja nie jest ustawiana, nawet jeśli subskrypcja została utworzona przy użyciu tej opcji.

Jeśli ta opcja nie jest używana, poprzednio zachowane komunikaty są wysyłane do podanej kolejki docelowej. Jeśli to działanie nie powiedzie się z powodu błędu (MQRC_RETAINED_MSG_Q_ERROR lub MQRC_RETAINED_NOT_DOSTARCZONE), utworzenie subskrypcji nie powiedzie się.

Publikacje MQSO_ON_REQUEST
Ustawienie tej opcji wskazuje, że subskrybent będzie w razie potrzeby żądawał informacji. Menedżer kolejek nie wysyła niezamówionych komunikatów do subskrybenta. Zachowana publikacja (lub być może wiele publikacji, jeśli w temacie określono znak wieloznaczny) jest wysyłana do subskrybenta za każdym razem, gdy wywołanie MQSUBRQ jest wykonywane przy użyciu uchwytu Hsub z poprzedniego wywołania MQSUB. Przy użyciu tej opcji nie są wysyłane żadne publikacje w wyniku wywołania MQSUB. W przypadku powrotu z wywołania MQSUB używającego opcji MQSO_RESUME ta opcja jest ustawiana w razie potrzeby.

Ta opcja nie jest poprawna w połączeniu z opcją SubLevel większą niż 1.

Opcje odczytu z wyprzedzeniem: Następujące opcje sterują tym, czy komunikaty nietrwałe są wysyłane do aplikacji przed aplikacją żądającą tych komunikatów.

MQSO_READ_AHEAD_AS_Q_DEF,
Jeśli wywołanie MQSUB używa zarządzanego uchwytu, domyślny atrybut odczytu z wyprzedzeniem kolejki modelowej powiązanej z subskrybowanym tematem określa, czy komunikaty są wysyłane do aplikacji przed zażądaniem ich przez aplikację.

Jest to wartość domyślna.

MQSO_NO_READ_AHEAD
Jeśli wywołanie MQSUB używa zarządzanego uchwytu, komunikaty nie są wysyłane do aplikacji, zanim aplikacja ich zażąda.

MQSO_READ_AHEAD
Jeśli wywołanie MQSUB używa zarządzanego uchwytu, komunikaty mogą zostać wysłane do aplikacji przed zażądaniem ich przez aplikację.

Uwaga:

Poniższe uwagi dotyczą opcji odczytu z wyprzedzeniem:

1. Można podać tylko jedną z tych opcji. Jeśli określono zarówno MQOO_READ_AHEAD, jak i MQOO_NO_READ_AHEAD, zwracany jest kod przyczyny MQRC_OPTIONS_ERROR. Te opcje mają zastosowanie tylko wtedy, gdy określono opcję MQSO_MANAGED.
2. Nie mają one zastosowania do MQSUB, gdy przekazywana jest kolejka, która została wcześniej otwarta. Odczyt z wyprzedzeniem może nie być włączony na żądanie. Opcje MQGET użyte w pierwszym wywołaniu MQGET mogą uniemożliwić włączenie odczytu z wyprzedzeniem. Ponadto odczyt z wyprzedzeniem jest wyłączany, gdy klient łączy się z menedżerem kolejek, w którym odczyt z wyprzedzeniem nie jest obsługiwany. Jeśli aplikacja nie działa jako klient IBM MQ , te opcje są ignorowane.

Opcje znaków wieloznacznych: Następujące opcje sterują sposobem interpretowania znaków wieloznacznych w łańcuchu podanym w polu ObjectString pliku MQSD. Można określić tylko jedną

z tych opcji. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER nie można zmieniać tych opcji znaków wieloznacznych. Po powrocie z wywołania MQSUB z użyciem opcji MQSO_RESUME ustawiana jest odpowiednia opcja znaku wieloznacznego.

MQSO_WILDCARD_CHAR

Znaki wieloznaczne działają tylko na znakach w łańcuchu tematu.

Zachowanie zdefiniowane przez MQSO_WILDCARD_CHAR przedstawiono w poniższej tabeli.

Tabela 528. Interpretowanie znaków wieloznacznych

Znak specjalny	**zachowanie;**
Prawy ukośnik (/)	Bez znaczenia, tylko inny znak
Gwiazdka (*)	Znak wieloznaczny, zero lub więcej znaków
Znak zapytania (?)	Znak wieloznaczny, 1 znak
Znak procentu (%)	Znak zmiany znaczenia, który umożliwia użycie znaków (*), (?) lub (%) w łańcuchu i nie jest interpretowany jako znak specjalny, na przykład (% *), (%?) lub (%%).

Na przykład publikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

jest zgodne z subskrybentami przy użyciu następujących tematów:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Uwaga: Użycie znaków wieloznacznych podczas używania komunikatów w formacie MQRFH1 do publikowania/subskrypcji jest dokładnie takie samo znaczenie, jak w produkcie IBM MQ V6 i WebSphere MB V6 . Zaleca się, aby ta opcja nie była używana w przypadku nowo napisanych aplikacji i była używana tylko w przypadku aplikacji, które były wcześniej uruchamiane dla tej wersji i nie zostały zmienione w celu użycia domyślnego zachowania znaku wieloznacznego zgodnie z opisem w sekcji MQSO_WILDCARD_TOPIC.

MQSO_WILDCARD_TOPIC,

Znaki wieloznaczne działają tylko na elementach tematu w łańcuchu tematu. Jest to zachowanie domyślne, jeśli nie zostanie wybrana żadna opcja.

Zachowanie wymagane przez parametr MQSO_WILDCARD_TOPIC jest przedstawione w poniższej tabeli:

Tabela 529. Interpretowanie znaków wieloznacznych

Znak specjalny	**zachowanie;**
(/)	Separator poziomu tematu
Znak numeru (#)	Znak wieloznaczny: poziom wielu tematów
Znak plusa (+)	Znak wieloznaczny: poziom pojedynczego tematu

Uwagi:

Znaki (+) i (#) nie są traktowane jako znaki wieloznaczne, jeśli są mieszane z innymi znakami (w tym z samymi sobą) na poziomie tematu. W poniższym łańcuchu znaki (#) i (+) są traktowane jako zwykłe znaki.

```
level0/level1/#+/level3/level#
```

Na przykład publikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

jest zgodne z subskrybentami przy użyciu następujących tematów:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Inne opcje: Następujące opcje sterują sposobem wywoływania interfejsu API, a nie subskrypcją. Po powrocie z wywołania MQSUB używającego komendy MQSO_RESUME te opcje nie ulegają zmianie. Więcej informacji na temat zawiera sekcja "AlternateUserId (MQCHAR12) dla MQSD" na stronie 599.

MQSO_ALTERNATE_UPRAWNIENIA_UŻYTKOWNIKA

Pole identyfikatora AlternateUserzawiera identyfikator użytkownika używany do sprawdzania poprawności tego wywołania MQSUB. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy ten identyfikator AlternateUserma uprawnienia do otwierania obiektu z określonymi opcjami dostępu, bez względu na to, czy identyfikator użytkownika, pod którym działa aplikacja, jest do tego uprawniony.

MQSO_SET_CORREL_ID (Identyfikator korelacji MQSO_SET_)

Subskrypcja ma używać identyfikatora korelacji podanego w polu *SubCorrelId* . Jeśli ta opcja nie zostanie podana, identyfikator korelacji zostanie automatycznie utworzony przez menedżer kolejek w czasie subskrypcji i zwrócony do aplikacji w polu *SubCorrelId* . Więcej informacji na ten temat zawiera sekcja "SubCorrelId (MQBYTE24) dla MQSD" na stronie 601 .

Tej opcji nie można łączyć z opcją MQSO_MANAGED.

MQSO_SET_IDENTITY_CONTEXT,

Subskrypcja ma używać tokenu rozliczania i danych tożsamości aplikacji podanych w polach *PubAccountingToken* i *PubApplIdentityData* .

Jeśli ta opcja jest określona, wykonywane jest takie samo sprawdzenie autoryzacji, jak w przypadku dostępu do kolejki docelowej za pomocą wywołania MQOPEN z opcją MQOO_SET_IDENTITY_CONTEXT, z wyjątkiem sytuacji, gdy opcja MQSO_MANAGED jest również używana, w której to przypadku nie ma sprawdzania autoryzacji w kolejce docelowej.

Jeśli ta opcja nie jest określona, publikacje wysyłane do tego subskrybenta mają powiązane domyślne informacje o kontekście w następujący sposób:

Tabela 530. Domyślne informacje o kontekście dla publikacji wysyłanych do tego subskrybenta

Pole w strukturze MQMD	**Użyta wartość**
UserIdentifier	Identyfikator użytkownika powiązany z subskrypcją w momencie jej wykonywania.
AccountingToken	Określony na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie należy ustawić wartość MQACT_NONE.
ApplIdentityData	Ustaw na wartości puste

Ta opcja jest poprawna tylko z opcjami MQSO_CREATE i MQSO_ALTER. W przypadku użycia z opcją MQSO_RESUME pola *PubAccountingToken* i *PubApplIdentityData* są ignorowane, więc ta opcja nie ma zastosowania.

Jeśli subskrypcja zostanie zmieniona bez użycia tej opcji, gdy wcześniej informacje o kontekście tożsamości zostały dostarczone przez subskrypcję, dla zmienionej subskrypcji zostaną wygenerowane domyślne informacje o kontekście.

Jeśli subskrypcja zezwalająca różnym identyfikatorom użytkowników na korzystanie z niej z opcją MQSO_ANY_USERID jest wznawiana przez inny ID użytkownika, dla nowego ID użytkownika będącego właścicielem subskrypcji generowany jest domyślny kontekst tożsamości, a wszystkie kolejne publikacje są dostarczane z nowym kontekstem tożsamości.

MQSO_FAIL_IF_QUIESCING,

Wywołanie MQSUB kończy się niepowodzeniem, jeśli menedżer kolejek jest w stanie wyciszania. W systemie z/OSw przypadku aplikacji CICS lub IMS ta opcja wymusza także niepowodzenie wywołania MQSUB, jeśli połączenie jest w stanie wyciszania.

ObjectName (MQCHAR48) dla usługi MQSD

Jest to nazwa obiektu tematu zdefiniowana w lokalnym menedżerze kolejek.

Nazwa może zawierać następujące znaki:

- Wielkie litery (od A do Z)
- Małe litery (od a do z)
- Cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani wewnętrznych odstępów, ale może zawierać końcowe odstępy. Znak o kodzie zero oznacza koniec istotnych danych w nazwie; znak o kodzie zero i wszystkie następujące po nim znaki są traktowane jako odstępy. W podanych środowiskach obowiązują następujące ograniczenia:

- W systemach używających kodu EBCDIC Katakana nie można używać małych liter.
- W systemie z/OS:
  - Należy unikać nazw, które zaczynają się lub kończą znakiem podkreślenia; nie mogą być przetwarzane przez operacje i panele sterowania.
  - Znak procentu ma specjalne znaczenie dla RACF. Jeśli produkt RACF jest używany jako zewnętrzny menedżer zabezpieczeń, nazwy nie mogą zawierać procentu. Jeśli tak, nazwy te nie są uwzględniane podczas sprawdzania zabezpieczeń, gdy używane są profile ogólne RACF .
- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w cudzysłów, jeśli zostały podane w komendach. Te cudzysłowy nie mogą być podawane dla nazw, które występują jako pola w strukturach lub jako parametry w wywołaniach.

Nazwa *ObjectName* jest używana do utworzenia pełnej nazwy tematu.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

Jeśli nie można znaleźć obiektu identyfikowanego przez pole *ObjectName* , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_UNKNOWN_OBJECT_NAME, nawet jeśli w parametrze *ObjectString*podano łańcuch.

Po powrocie z wywołania MQSUB z opcją MQSO_RESUME to pole pozostaje niezmienione.

Długość tego pola jest określona przez wartość MQ_TOPIC_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER nie można zmienić nazwy obiektu tematu, który został zasubskrybowany. To pole i pole *ObjectString* można pominąć. Jeśli zostaną podane, muszą zostać przetłumaczone na tę samą pełną nazwę tematu. Jeśli nie, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_TOPIC_NOT_ALTERABLE.

AlternateUserId (MQCHAR12) dla MQSD

Jeśli zostanie podana wartość MQSO_ALTERNATE_USER_AUTHORITY, to pole zawiera alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji subskrypcji i danych wyjściowych do kolejki docelowej (określonej w parametrze **Hobj** wywołania MQSUB) zamiast identyfikatora użytkownika, który jest aktualnie używany przez aplikację.

Jeśli operacja powiedzie się, identyfikator użytkownika określony w tym polu jest rejestrowany jako identyfikator użytkownika będącego właścicielem subskrypcji zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona.

Jeśli określono parametr MQSO_ALTERNATE_USER_AUTHORITY i to pole jest całkowicie puste, aż do pierwszego znaku o wartości NULL lub końca pola, subskrypcja może zakończyć się powodzeniem tylko wtedy, gdy do subskrybowania tego tematu z określonymi opcjami lub kolejką docelową dla danych wyjściowych nie jest wymagana autoryzacja użytkownika.

Jeśli opcja MQSO_ALTERNATE_USER_AUTHORITY nie jest określona, to pole jest ignorowane.

W podanych środowiskach występują następujące różnice:

- W systemie z/OSdo sprawdzenia autoryzacji subskrypcji używane jest tylko 8 pierwszych znaków identyfikatora AlternateUser. Jednak bieżący identyfikator użytkownika musi być autoryzowany do określenia tego konkretnego alternatywnego identyfikatora użytkownika; do tego sprawdzenia używane są wszystkie 12 znaków alternatywnego identyfikatora użytkownika. Identyfikator użytkownika może zawierać tylko znaki dozwolone przez zewnętrznego menedżera zabezpieczeń.

Po powrocie z wywołania MQSUB używającego komendy MQSO_RESUME to pole pozostaje niezmienione.

Jest to pole wejściowe. Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 12 znaków odstępu w innych językach programowania.

AlternateSecurityId (MQBYTE40) dla MQSD

Jest to identyfikator bezpieczeństwa, który jest przekazywany z identyfikatorem AlternateUserdo usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji.

AlternateSecurityIdentyfikator jest używany tylko wtedy, gdy określono wartość MQSO_ALTERNATE_USER_AUTHORITY, a pole identyfikatora AlternateUsernie jest całkowicie puste aż do pierwszego znaku o kodzie zero lub końca pola.

Po powrocie z wywołania MQSUB używającego komendy MQSO_RESUME to pole pozostaje niezmienione.

Więcej informacji zawiera opis parametru “AlternateSecurityId (MQBYTE40) dla MQOD” na stronie 505 w typie danych MQOD.

SubExpiry (MQLONG) dla MQSD

Jest to czas wyrażony w dziesiątych częściach sekundy, po upływie którego subskrypcja traci ważność. Po upływie tego okresu żadne publikacje nie będą zgodne z tą subskrypcją. Gdy tylko subskrypcja utraci ważność, publikacje nie będą już wysyłane do kolejki. Jednak publikacje, które już istnieją, nie są w żaden sposób naruszone. *SubExpiry* nie ma wpływu na utratę ważności publikacji.

Rozpoznawana jest następująca wartość specjalna:

MQEI_UNLIMITED,
Subskrypcja ma nieograniczony czas ważności.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER można zmienić datę ważności subskrypcji.

Po powrocie z wywołania MQSUB przy użyciu opcji MQSO_RESUME w tym polu jest ustawiana pierwotna utrata ważności subskrypcji, a nie pozostały czas utraty ważności.

ObjectString (MQCHARV) dla MQSD

Jest to długa nazwa obiektu, która ma być używana.

Nazwa *ObjectString* jest używana do tworzenia pełnej nazwy tematu.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ObjectName* i *ObjectString*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

Maksymalna długość łańcucha *ObjectString* wynosi 10240.

Jeśli parametr *ObjectString* nie jest określony poprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_STRING_ERROR.

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

Jeśli w pliku *ObjectString* występują znaki wieloznaczne, interpretacja tych znaków może być kontrolowana za pomocą opcji znaków wieloznacznych określonych w polu Opcje w pliku MQSD.

Po powrocie z wywołania MQSUB z opcją MQSO_RESUME to pole pozostaje niezmienione. Pełna nazwa tematu jest zwracana w polu *ResObjectString* , jeśli podano bufor.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER nie można zmienić długiej nazwy obiektu tematu, który został zasubskrybowany. To pole i pole *ObjectName* można pominąć. Jeśli zostaną podane, muszą zostać przetłumaczone na tę samą pełną nazwę tematu, w przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony komunikat MQRC_TOPIC_NOT_ALTERABLE.

SubName (MQCHARV) dla MQSD

Określa nazwę subskrypcji. To pole jest wymagane tylko wtedy, gdy parametr *Options* określa opcję MQSO_DURABLE, ale jeśli zostanie podany, będzie używany przez menedżer kolejek również dla opcji MQSO_NON_DURABLE.

Jeśli parametr *SubName* jest określony, musi być unikalny w obrębie menedżera kolejek, ponieważ jest to metoda używana do identyfikowania subskrypcji.

Maksymalna długość łańcucha *SubName* wynosi 10240.

To pole służy dwóm celom. W przypadku subskrypcji MQSO_DURABLE to pole służy do identyfikowania subskrypcji, aby można było ją wznowić po jej utworzeniu, jeśli uchwyt subskrypcji został zamknięty (za pomocą opcji MQCO_KEEP_SUB) lub został odłączony od menedżera kolejek. W tym celu należy użyć wywołania MQSUB z opcją MQSO_RESUME. Jest on również wyświetlany w widoku administracyjnym subskrypcji w polu SUBID w kolumnie DISPLAY SBSTATUS.

Jeśli parametr *SubName* zostanie określony niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV , zostanie ona pozostawiona, gdy jest wymagana (czyli *SubName*). *VSLength* ma wartość zero) lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SUB_NAME_ERROR.

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER nie można zmienić nazwy subskrypcji, ponieważ jest to pole identyfikujące używane do znajdowania przywoływanej subskrypcji. Nie jest on zmieniany w danych wyjściowych wywołania MQSUB z opcją MQSO_RESUME.

Dane SubUser(MQCHARV) dla MQSD

Określa dane użytkownika subskrypcji. Dane udostępnione w subskrypcji w tym polu zostaną dołączone jako właściwość komunikatu danych MQSubUserdla każdej publikacji wysyłanej do tej subskrypcji.

Maksymalna długość łańcucha *SubUserData* wynosi 10240.

Jeśli parametr *SubUserData* zostanie podany niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekroczy ona maksymalną długość, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_SUB_USER_DATA_ERROR.

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER można zmienić dane użytkownika subskrypcji.

To pole o zmiennej długości jest zwracane w danych wyjściowych wywołania MQSUB przy użyciu opcji MQSO_RESUME, jeśli podano bufor i w pliku *VSBufLen*występuje dodatnia długość buforu. Jeśli w wywołaniu nie podano buforu, w polu *VSLength* komendy MQCHARV zwracana jest tylko długość daty użytkownika subskrypcji. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w podanym buforze zwracane są tylko *VSBufLen* bajtów.

SubCorrelId (MQBYTE24) dla MQSD

To pole zawiera identyfikator korelacji wspólny dla wszystkich publikacji zgodnych z tą subskrypcją.

Ostrzeżenie: Identyfikator korelacji może być przekazywany tylko między menedżerami kolejek w klastrze publikowania/subskrybowania, a nie w hierarchii.

Wszystkie publikacje wysłane w celu dopasowania tej subskrypcji zawierają ten identyfikator korelacji w deskryptorze komunikatu. Jeśli wiele subskrypcji pobiera swoje publikacje z tej samej kolejki, użycie komendy MQGET według identyfikatora korelacji umożliwia uzyskanie tylko publikacji dla konkretnej subskrypcji. Ten identyfikator korelacji może zostać wygenerowany przez menedżera kolejek lub przez użytkownika.

Jeśli opcja MQSO_SET_CORREL_ID nie jest określona, identyfikator korelacji jest generowany przez menedżer kolejek, a to pole jest polem wyjściowym zawierającym identyfikator korelacji, który zostanie ustawiony w każdym komunikacie opublikowanym dla tej subskrypcji. Wygenerowany identyfikator korelacji składa się z 4-bajtowego identyfikatora produktu (AMQX lub CSQM w kodzie ASCII lub EBCDIC), po którym następuje specyficzna dla produktu implementacja unikalnego łańcucha.

Jeśli podano opcję MQSO_SET_CORREL_ID, identyfikator korelacji jest generowany przez użytkownika, a to pole jest polem wejściowym zawierającym identyfikator korelacji, który ma zostać ustawiony w każdej publikacji dla tej subskrypcji. W tym przypadku, jeśli pole zawiera wartość MQCI_NONE, identyfikator korelacji, który jest ustawiany w każdym komunikacie publikowanym dla tej subskrypcji, jest identyfikatorem korelacji utworzonym przez oryginalne umieszczenie komunikatu.

Jeśli podano opcję MQSO_GROUP_SUB, a określony identyfikator korelacji jest taki sam, jak istniejąca zgrupowana subskrypcja używająca tej samej kolejki i nakładającego się łańcucha tematu, z kopią publikacji udostępniana jest tylko najbardziej znacząca subskrypcja w grupie.

Długość tego pola jest określona przez wartość MQ_CORREL_ID_LENGTH. Wartością początkową tego pola jest MQCI_NONE.

Jeśli istniejąca subskrypcja jest modyfikowana za pomocą opcji MQSO_ALTER, a to pole jest polem wejściowym, identyfikator korelacji subskrypcji może zostać zmieniony, chyba że subskrypcja jest subskrypcją grupową, czyli została utworzona za pomocą opcji MQSO_GROUP_SUB, w której to przypadku nie można zmienić identyfikatora korelacji subskrypcji.

W przypadku powrotu z wywołania MQSUB używającego opcji MQSO_RESUME to pole jest ustawiane na bieżący identyfikator korelacji dla subskrypcji.

PubPriority (MQLONG) dla MQSD

Jest to wartość znajdująca się w polu *Priority* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. Więcej informacji na temat pola *Priority* w strukturze MQMD zawiera sekcja "Priorytet (MQLONG) dla MQMD" na stronie 463.

Wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następujących wartości specjalnych:

MQPRI_PRIORITY_AS_Q_DEF

Jeśli kolejka subskrypcji jest podana w polu *Hobj* wywołania MQSUB i nie jest zarządzanym uchwytem, priorytet dla komunikatu jest pobierany z atrybutu **DefPriority** tej kolejki. Jeśli kolejka jest kolejką klastra lub istnieje więcej niż jedna definicja w ścieżce rozstrzygania nazw kolejek, priorytet jest określany, gdy komunikat publikacji jest umieszczany w kolejce zgodnie z opisem w sekcji "Priorytet (MQLONG) dla MQMD" na stronie 463.

Jeśli wywołanie MQSUB używa zarządzanego uchwytu, priorytet komunikatu jest pobierany z atrybutu **DefPriority** kolejki modelowej powiązanej z subskrybowanym tematem.

MQPRI_PRIORITY_AS_PUBLISHED

Priorytet komunikatu jest priorytetem oryginalnej publikacji. Jest to wartość początkowa pola.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER można zmienić wartość parametru *Priority* dla wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB używającego opcji MQSO_RESUME to pole jest ustawiane na bieżący priorytet używany dla subskrypcji.

Znacznik PubAccounting(MQBYTE32) dla usługi MQSD

Jest to wartość znajdująca się w polu *AccountingToken* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *AccountingToken* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu. Więcej informacji na temat pola *AccountingToken* w strukturze MQMD zawiera sekcja "AccountingToken (MQBYTE32) dla MQMD" na stronie 472

W polu *PubAccountingToken* można użyć następującej wartości specjalnej:

MQACT_NONE

Nie określono tokenu rozliczania.

Wartością długości pola jest zero binarne.

Dla języka programowania C zdefiniowana jest również stała MQACT_NONE_ARRAY, która ma taką samą wartość jak MQACT_NONE, ale jest tablicą znaków zamiast łańcucha.

Jeśli opcja MQSO_SET_IDENTITY_CONTEXT nie jest określona, znacznik rozliczania jest generowany przez menedżer kolejek jako domyślna informacja o kontekście, a to pole jest polem wyjściowym zawierającym element *AccountingToken* , który zostanie ustawiony w każdym komunikacie publikowanym dla tej subskrypcji.

Jeśli określono opcję MQSO_SET_IDENTITY_CONTEXT, znacznik rozliczania jest generowany przez użytkownika, a to pole jest polem wejściowym zawierającym element *AccountingToken* , który ma zostać ustawiony w każdej publikacji dla tej subskrypcji.

Długość tego pola jest określona przez parametr MQ_ACCOUNTING_TOKEN_LENGTH. Wartością początkową tego pola jest MQACT_NONE.

W przypadku zmiany istniejącej subskrypcji za pomocą opcji MQSO_ALTER można zmienić wartość parametru *AccountingToken* w przyszłych komunikatach publikacji.

W przypadku powrotu z wywołania MQSUB używającego opcji MQSO_RESUME to pole jest ustawiane na bieżącą wartość parametru *AccountingToken* używanego na potrzeby subskrypcji.

PubApplIdentityData (MQCHAR32) dla MQSD

Jest to wartość znajdująca się w polu *ApplIdentityData* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *ApplIdentityData* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu. Więcej informacji na temat pola *ApplIdentityData* w strukturze MQMD zawiera sekcja “Dane ApplIdentity(MQCHAR32) dla MQMD” na stronie 473

Jeśli opcja MQSO_SET_IDENTITY_CONTEXT nie zostanie określona, parametr *ApplIdentityData* , który jest ustawiany w każdym komunikacie publikowanym dla tej subskrypcji, będzie pusty jako domyślna informacja o kontekście.

Jeśli podano opcję MQSO_SET_IDENTITY_CONTEXT, plik *PubApplIdentityData* jest generowany przez użytkownika i to pole jest polem wejściowym zawierającym plik *ApplIdentityData* , który ma zostać ustawiony w każdej publikacji dla tej subskrypcji.

Długość tego pola jest określona przez wartość MQ_APPL_IDENTITY_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 32 znaki odstępu w innych językach programowania.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji MQSO_ALTER można zmienić wartość parametru *ApplIdentityData* dla wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB używającego opcji MQSO_RESUME to pole jest ustawiane na bieżącą wartość parametru *ApplIdentityData* używanego na potrzeby subskrypcji.

SelectionString (MQCHARV) dla usługi MQSD

Jest to łańcuch używany do udostępniania kryteriów wyboru używanych podczas subskrybowania komunikatów z tematu.

To pole o zmiennej długości będzie zwracane w danych wyjściowych wywołania MQSUB przy użyciu opcji MQSO_RESUME (jeśli podano bufor), a także w polu VSBufSizeznajduje się dodatnia długość buforu. Jeśli w wywołaniu nie podano buforu, w polu VSLength tabeli MQCHARV zostanie zwrócona tylko długość łańcucha wyboru. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w podanym buforze są zwracane tylko bajty VSBufSize .

Jeśli parametr *SelectionString* jest określony niepoprawnie, zgodnie z opisem sposobu użycia struktury “MQCHARV-łańcuch o zmiennej długości” na stronie 299 lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_SELECTION_STRING_ERROR.

Składnia SelectionString jest opisana w sekcji Selektory.

SubLevel (MQLONG) dla MQSD

Jest to poziom powiązany z subskrypcją. Publikacje są dostarczane do tej subskrypcji tylko wtedy, gdy znajduje się ona w zestawie subskrypcji o najwyższej wartości SubLevel , która jest mniejsza lub równa wartości PubLevel użytej w czasie publikacji. Jeśli jednak publikacja została zachowana, nie jest już dostępna dla subskrybentów na wyższych poziomach, ponieważ jest ponownie publikowana na poziomie PubLevel 1.

Wartość musi być z zakresu od 0 do 9. Zero jest najniższym poziomem.

Wartością początkową tego pola jest 1.

Więcej informacji na ten temat zawiera sekcja Przechwytywanie publikacji.

Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji MQSO_ALTER, nie można zmienić wartości SubLevel .

Łączenie SubLevel z wartością większą niż 1 z opcją MQSO_PUBLIC ATIONS_ON_REQUEST nie jest dozwolone.

Po powrocie z wywołania MQSUB używającego opcji MQSO_RESUME to pole jest ustawiane na bieżący poziom używany dla subskrypcji.

ResObjectŁańcuch (MQCHARV) dla MQSD

Jest to długa nazwa obiektu po tym, jak menedżer kolejek rozstrzyga nazwę podaną w programie *ObjectName*.

Jeśli w polu *ObjectString* zostanie podana długa nazwa obiektu i w polu *ObjectName*nie zostanie podana żadna wartość, wartość zwracana w tym polu będzie taka sama, jak w polu *ObjectString*.

Jeśli to pole zostanie pominięte (czyli pole ResObjectString.VSBufSize ma wartość zero), pole *ResObjectString* nie zostanie zwrócone, ale długość zostanie zwrócona w polu ResObjectString.VSLength. Jeśli długość jest krótsza niż pełny łańcuch ResObject, jest ona obcinana i zwraca tyle znaków po prawej stronie, ile może zmieścić się w podanej długości.

Jeśli parametr *ResObjectString* zostanie podany niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekroczy ona maksymalną długość, wywołanie nie powiedzie się z kodem przyczyny MQRC_RES_OBJECT_STRING_ERROR.

MQSMPO-ustawianie opcji właściwości komunikatu

Struktura **MQSMPO** umożliwia aplikacjom określanie opcji sterujących ustawianiem właściwości komunikatów. Struktura jest parametrem wejściowym wywołania **MQSETMP** .

Dostępność

Wszystkie systemy IBM MQ i klienty IBM MQ .

Zestaw znaków i kodowanie

Dane w pliku **MQSMPO** muszą znajdować się w zestawie znaków aplikacji i kodowania aplikacji (**MQENC_NATIVE**).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 531. Pola w MQSMPO

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQSMPO_STRUC_ID	'SMPO'
Wersja (numer wersji struktury)	MQSMPO_VERSION_1	1
Opcje (opcje)	MQSMPO_BRAK	0
ValueEncoding (kodowanie wartości właściwości)	RODZIMA MQENC	Zależy od środowiska
ValueCCSID (zestaw znaków wartości właściwości)	APPL MQCCSI_PL	-3

Uwagi:

1. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
2. W języku programowania C: zmienna makraMQSMPO_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQSMPO MySMPO = {MQSMPO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQSMPO

```
typedef struct tagMQSMPO MQSMPO;
struct tagMQSMPO {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    Options;         /* Options that control the action of MQSETMP */
  MQLONG    ValueEncoding;   /* Encoding of Value */
  MQLONG    ValueCCSID;      /* Character set identifier of Value */
};
```

Deklaracja języka COBOL dla MQSMPO

```
**   MQSMPO structure
  10 MQSMPO.
**    Structure identifier
   15 MQSMPO-STRUCID       PIC X(4).
**    Structure version number
   15 MQSMPO-VERSION       PIC S9(9) BINARY.
**    Options that control the action of MQSETMP
   15 MQSMPO-OPTIONS       PIC S9(9) BINARY.
**    Encoding of VALUE
   15 MQSMPO-VALUEENCODING PIC S9(9) BINARY.
**    Character set identifier of VALUE
   15 MQSMPO-VALUECCSID    PIC S9(9) BINARY.
```

Deklaracja PL/I dla MQSMPO

```
dcl
 1 MQSMPO based,
   3 StrucId        char(4),       /* Structure identifier */
   3 Version        fixed bin(31), /* Structure version number */
   3 Options        fixed bin(31), /* Options that control the action of MQSETMP */
   3 ValueEncoding  fixed bin(31), /* Encoding of Value */
   3 ValueCCSID     fixed bin(31), /* Character set identifier of Value */
```

Deklaracja High Level Assembler dla MQSMPO

```
MQSMPO                DSECT
MQSMPO_STRUCID        DS   CL4   Structure identifier
MQSMPO_VERSION        DS   F     Structure version number
MQSMPO_OPTIONS        DS   F     Options that control the action of
*                                MQSETMP
MQSMPO_VALUEENCODING  DS   F     Encoding of VALUE
MQSMPO_VALUECCSID     DS   F     Character set identifier of VALUE
MQSMPO_LENGTH         EQU  *-MQSMPO
MQSMPO_AREA           DS   CL(MQSMPO_LENGTH)
```

StrucId (MQCHAR4) dla MQSMPO

Jest to identyfikator struktury struktury ustawionej struktury opcji właściwości komunikatu. Jest to zawsze pole wejściowe. Jego wartością jest MQSMPO_STRUC_ID.

Wartość musi być następująca:

MQSMPO_STRUC_ID

Identyfikator struktury opcji ustawiania właściwości komunikatu.

Dla języka programowania C zdefiniowana jest również stała **MQSMPO_STRUC_ID_ARRAY** . Ma taką samą wartość jak **MQSMPO_STRUC_ID**, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQSMPO

Jest to numer wersji struktury; wartość musi być następująca:

MQSMPO_VERSION_1
Version-1 służy do ustawiania struktury opcji właściwości komunikatu.

Następująca stała określa numer wersji bieżącej:

MQSMPO_CURRENT_VERSION
Bieżąca wersja struktury ustawionych opcji właściwości komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **MQSMPO_VERSION_1**.

Opcje (MQLONG) dla MQSMPO

Opcje lokalizacji

Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości:

MQSMPO_SET_FIRST
Ustawia wartość pierwszej właściwości, która jest zgodna z podaną nazwą, lub, jeśli nie istnieje, dodaje nową właściwość po wszystkich innych właściwościach ze zgodną hierarchią.

MQSMPO_SET_PROP_UNDER_CURSOR
Ustawia wartość właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości jest tą, do której ostatnio wysłano zapytanie przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany w wywołaniu MQGET lub gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze ustawiony lub jeśli wskaźnik właściwości wskazywany przez kursor właściwości został usunięty, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_BEFORE_CURSOR,
Ustawia nową właściwość przed właściwością wskazywanym przez kursor właściwości. Właściwość wskazywana przez kursor właściwości jest tą, do której ostatnio wysłano zapytanie przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany w wywołaniu MQGET lub gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze ustawiony lub jeśli wskaźnik właściwości wskazywany przez kursor właściwości został usunięty, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_SET_PROP_AFTER_CURSOR,
Ustawia nową właściwość po właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości jest tą, do której ostatnio wysłano zapytanie przy użyciu opcji MQIMPO_INQ_FIRST lub MQIMPO_INQ_NEXT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany w wywołaniu MQGET lub gdy uchwyt komunikatu jest określony w polu *MsgHandle* struktury MQGMO lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze ustawiony lub jeśli wskaźnik właściwości wskazywany przez kursor właściwości został usunięty, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

MQSMPO_APPEND_PROPERTY
Powoduje dodanie nowej właściwości po wszystkich innych właściwościach ze zgodną hierarchią. Jeśli istnieje co najmniej jedna właściwość, która jest zgodna z podaną nazwą, na końcu listy właściwości dodawana jest nowa właściwość.

Ta opcja umożliwia utworzenie listy właściwości o takiej samej nazwie.

Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

MQSMPO_BRAK
Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQSMPO_SET_FIRST.

ValueEncoding (MQLONG) dla MQSMPO

Kodowanie wartości właściwości, która ma zostać ustawiona, jeśli wartość jest wartością liczbową.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **MQENC_NATIVE**.

ValueCCSID (MQLONG) dla MQSMPO

Zestaw znaków wartości właściwości, który ma zostać ustawiony, jeśli wartość jest łańcuchem znaków.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **MQCCSI_APPL**.

MQSRO-opcje żądania subskrypcji

Struktura MQSRO umożliwia aplikacji określenie opcji sterujących sposobem wysyłania żądań subskrypcji. Struktura jest parametrem wejścia/wyjścia wywołania MQSUBRQ.

Dostępność

Struktura MQSRO jest dostępna na następujących platformach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

i dla IBM MQ MQI clients połączonych z tymi systemami.

Wersja

Bieżąca wersja MQSRO to MQSRO_VERSION_1.

Zestaw znaków i kodowanie

Dane w obiekcie MQSRO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez właściwość MQENC_NATIVE. Jeśli jednak aplikacja działa jako klient MQI produktu MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQSRO_ID_struktury	'SRO¬'
Wersja (numer wersji struktury)	MQSRO_VERSION_1	1
Opcje (opcje)	MQSRO_BRAK	0
NumPubs (liczba publikacji)	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQSRO_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQSRO MySRO = {MQSRO_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQSRO

```
typedef struct tagMQSRO MQSRO;
struct tagMQSRO {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    Options;              /* Options that control the action of MQSUBRQ */
  MQLONG    NumPubs;              /* Number of publications sent */
  /* Ver:1 */
};
```

Deklaracja języka COBOL dla MQSRO

```
** MQSRO structure
 10  MQSRO.
** Structure identifier
 15  MQSRO-STRUCID             PIC X(4).
** Structure version number
 15  MQSRO-VERSION             PIC S9(9) BINARY.
** Options that control the action of MQSUBRQ
 15  MQSRO-OPTIONS             PIC S9(9) BINARY.
** Number of publications sent
 15  MQSRO-NUMPUBS             PIC S9(9) BINARY.
```

Deklaracja języka PL/I dla obiektu MQSRO

```
dcl
 1 MQSRO based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
  3 Options        fixed bin(31), /* Options that control the action of MQSUBRQ */
  3 NumPubs        fixed bin(31); /* Number of publications sent */
```

Deklaracja High Level Assembler dla MQSRO

```
MQSRO                   DSECT
MQSRO_STRUCID           DS    CL4   Structure identifier
MQSRO_VERSION           DS    F     Structure version number
```

```
MQSRO_OPTIONS            DS    F     Options that control the action of MQSUBRQ
MQSRO_NUMPUBS            DS    F     Number of publications sent
*
MQSRO_LENGTH             EQU   *-MQSRO
                         ORG   MQSRO
MQSRO_AREA               DS    CL(MQSRO_LENGTH)
```

StrucId (MQCHAR4) dla MQSRO

Jest to identyfikator struktury struktury opcji żądania subskrypcji. Jest to zawsze pole wejściowe. Jego wartością jest MQSRO_STRUC_ID.

Wartość musi być następująca:

MQSRO_ID_struktury
Identyfikator struktury opcji żądania subskrypcji.

Dla języka programowania C zdefiniowana jest również stała MQSRO_STRUC_ID_ARRAY. Ma taką samą wartość jak MQSRO_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQSRO

Jest to numer wersji struktury; wartość musi być następująca:

MQSRO_VERSION_1
Version-1 Struktura opcji żądania subskrypcji.

Następująca stała określa numer wersji bieżącej:

MQSRO_CURRENT_VERSION
Bieżąca wersja struktury opcji żądań subskrypcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MQSRO_VERSION_1.

Opcje (MQLONG) dla MQSRO

Należy podać jedną z następujących opcji. Można podać tylko jedną opcję.

MQSRO_FAIL_IF_QUIESCING,

Wywołanie MQSUBRQ nie powiedzie się, jeśli menedżer kolejek jest w stanie wyciszania. W systemie z/OSw przypadku aplikacji CICS lub IMS ta opcja wymusza również niepowodzenie wywołania MQSUBRQ, jeśli połączenie jest w stanie wyciszania.

Opcja domyślna: Jeśli opisana wcześniej opcja nie jest wymagana, należy użyć następującej opcji:

MQSRO_BRAK

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

MQSRO_NONE pomaga w dokumentacji programu. Chociaż ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ponieważ jej wartość wynosi zero, nie można jej wykryć.

NumPubs (MQLONG) dla MQSRO

Jest to pole wyjściowe zwracane do aplikacji w celu wskazania liczby publikacji wysłanych do kolejki subskrypcji w wyniku tego wywołania. Mimo że ta liczba publikacji została wysłana w wyniku tego wywołania, nie ma gwarancji, że ta liczba komunikatów będzie dostępna dla aplikacji, zwłaszcza jeśli są to komunikaty nietrwałe.

Jeśli subskrybujący temat zawiera znak wieloznaczny, może istnieć więcej niż jedna publikacja. Jeśli w łańcuchu tematu nie ma znaków wieloznacznych podczas tworzenia subskrypcji reprezentowanej przez *Hsub* , w wyniku tego wywołania zostanie wysłana co najwyżej jedna publikacja.

MQSTS-struktura raportowania statusu

Struktura MQSTS jest parametrem wyjściowym komendy MQSTAT. Komenda MQSTAT służy do pobierania informacji o statusie. Te informacje są zwracane w strukturze MQSTS.

Zestaw znaków i kodowanie

Dane znakowe w usłudze MQSTS znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut *CodedCharSetId* menedżera kolejek. Dane liczbowe w usłudze MQSTS są kodowane na komputerze rodzimym. Jest to określone w polu *Kodowanie*.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 532. Pola w MQSTS

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQSTS_STRUC_ID (ID struktury MQSTS)	'STAT¬'
Wersja (numer wersji struktury)	MQSTS_VERSION_1	1
CompCode (kod zakończenia pierwszego błędu)	MQCC_OK	0
Przyczyna (kod przyczyny pierwszego błędu)	MQRC_BRAK	0
PutSuccessLiczba (liczba pomyślnych asynchronicznych wywołań put)	Brak	0
PutWarningLiczba (liczba asynchronicznych wywołań umieszczania, w których wystąpiły ostrzeżenia)	Brak	0
PutFailure (liczba nieudanych asynchronicznych wywołań put)	Brak	0
ObjectType (typ obiektu, który uległ awarii)	MQOT_Q	1
ObjectName (nazwa obiektu, który uległ awarii)	Brak	Pusty łańcuch lub odstępy
ObjectQMgrNazwa (nazwa menedżera kolejek, do którego należy obiekt, który uległ awarii)	Brak	Pusty łańcuch lub odstępy
ResolvedObjectNazwa (rozstrzygnięta nazwa kolejki docelowej)	Brak	Pusty łańcuch lub odstępy
ResolvedQMgrNazwa (rozstrzygnięta nazwa docelowego menedżera kolejek)	Brak	Pusty łańcuch lub odstępy
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest wcześniejsza niż MQSTS_VERSION_2.		
ObjectString (długa nazwa obiektu, który uległ awarii)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
SubName (nazwa niesprawnej subskrypcji)	MQCHARV_DEFAULT	{NULL,0,0,0,-3}
OpenOptions (opcje otwarcia powiązane z awarią)	Brak	0
SubOptions(opcje subskrypcji powiązane z niepowodzeniem)	Brak	0

Tabela 532. Pola w MQSTS (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C zmienna makra MQSTS_DEFAULT zawiera wartości wymienione w tabeli. Można go użyć w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQSTS MySTS = {MQSTS_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQSTS

```
typedef struct tagMQSTS MQSTS;
struct tagMQSTS {
 MQCHAR4   StrucId;            /* Structure identifier */
 MQLONG    Version;            /* Structure version number */
 MQLONG    CompCode;           /* Completion Code of first error */
 MQLONG    Reason;             /* Reason Code of first error */
 MQLONG    PutSuccessCount;    /* Number of Async calls succeeded */
 MQLONG    PutWarningCount;    /* Number of Async calls had warnings */
 MQLONG    PutFailureCount;    /* Number of Async calls had failures */
 MQLONG    ObjectType;         /* Failing object type */
 MQCHAR48  ObjectName;         /* Failing object name */
 MQCHAR48  ObjectQMgrName;     /* Failing object queue manager name */
 MQCHAR48  ResolvedObjectName; /* Resolved name of destination queue */
 MQCHAR48  ResolvedQMgrName;   /* Resolved name of destination qmgr */
/* Ver:1 */
 MQCHARV   ObjectString;       /* Failing object long name */
 MQCHARV   SubName;            /* Failing subscription name */
 MQLONG    OpenOptions;        /* Failing open options */
 MQLONG    SubOptions;         /* Failing subscription options */
/* Ver:2 */
};
```

Deklaracja języka COBOL dla usługi MQSTS

```
      ** MQSTS structure
       10  MQSTS.
      ** Structure identifier
       15  MQSTS-STRUCID PIC X(4).
      ** Structure version number
       15  MQSTS-VERSION PIC S9(9) BINARY.
      ** Completion Code of first error
       15  MQSTS-COMPCODE PIC S9(9) BINARY.
      ** Reason Code of first error
       15  MQSTS-REASON PIC S9(9) BINARY.
      ** Number of Async put calls succeeded
       15  MQSTS-PUTSUCCESSCOUNT PIC S9(9) BINARY.
      ** Number of Async put calls had warnings
       15  MQSTS-PUTWARNINGCOUNT PIC S9(9) BINARY.
      ** Number of Async put calls had failures
       15  MQSTS-PUTFAILURECOUNT PIC S9(9) BINARY.
      ** Failing object type
       15  MQSTS-OBJECTTYPE PIC S9(9) BINARY.
      ** Failing object name
       15  MQSTS-OBJECTNAME PIC X(48).
      ** Failing object queue manager
       15  MQSTS-OBJECTQMGRNAME PIC X(48).
      ** Resolved name of destination queue
       15  MQSTS-RESOLVEDOBJECTNAME PIC X(48).
      ** Resolved name of destination qmgr
       15  MQSTS-RESOLVEDQMGRNAME PIC X(48).
```

```
      ** Ver:1 **
      ** Failing object long name
       15  MQSTS-OBJECTSTRING.
      ** Address of variable length string
       20  MQSTS-OBJECTSTRING-VSPTR POINTER.
      ** Offset of variable length string
       20  MQSTS-OBJECTSTRING-VSOFFSET PIC S9(9) BINARY.
      ** Size of buffer
       20  MQSTS-OBJECTSTRING-VSBUFSIZE PIC S9(9) BINARY.
      ** Length of variable length string
       20  MQSTS-OBJECTSTRING-VSLENGTH PIC S9(9) BINARY.
      ** CCSID of variable length string
       20  MQSTS-OBJECTSTRING-VSCCSID PIC S9(9) BINARY.
      ** Failing subscription name
       15  MQSTS-SUBNAME.
      ** Address of variable length string
       20  MQSTS-SUBNAME-VSPTR POINTER.
      ** Offset of variable length string
       20  MQSTS-SUBNAME-VSOFFSET PIC S9(9) BINARY.
      ** Size of buffer
       20  MQSTS-SUBNAME-VSBUFSIZE PIC S9(9) BINARY.
      ** Length of variable length string
       20  MQSTS-SUBNAME-VSLENGTH PIC S9(9) BINARY.
      ** CCSID of variable length string
       20  MQSTS-SUBNAME-VSCCSID PIC S9(9) BINARY.
      ** Failing open options
       15  MQSTS-OPENOPTIONS PIC S9(9) BINARY.
      ** Failing subscription options
       15  MQSTS-SUBOPTIONS PIC S9(9) BINARY.
      ** Ver:2 **
```

Deklaracja języka PL/I dla usługi MQSTS

```
dcl
   1 MQSTS based,
     3 StrucId             char(4),       /* Structure identifier */
      3 Version            fixed bin(31), /* Structure version number */
      3 CompCode           fixed bin(31), /* Completion code */
      3 Reason             fixed bin(31), /* Reason code */
      3 PutSuccessCount    fixed bin(31), /* Put success count */
      3 PutWarningCount    fixed bin(31), /* Put warning count */
      3 PutFailureCount    fixed bin(31), /* Put failure count */
      3 ObjectType         fixed bin(31), /* Object type */
      3 ObjectName         char(48), /* Object name */
      3 ObjectQmgrName     char(48), /* Object queue manager */
      3 ResolvedObjectName char(48), /* Resolved Object name */
      3 ResolvedQmgrName   char(48); /* Resolved Object queue manager */
  /* Ver:1 */
      3 ObjectString,                /* Failing object long name */
         5 VSPtr pointer,            /* Address of variable length string */
         5 VSOffset fixed bin(31),   /* Offset of variable length string */
         5 VSBufSize fixed bin(31),  /* Size of buffer */
         5 VSLength fixed bin(31),   /* Length of variable length string */
         5 VSCCSID fixed bin(31);    /* CCSID of variable length string */
      3 SubName,                     /* Failing subscription name */
         5 VSPtr pointer,            /* Address of variable length string */
         5 VSOffset fixed bin(31),   /* Offset of variable length string */
         5 VSBufSize fixed bin(31),  /* Size of buffer */
         5 VSLength fixed bin(31),   /* Length of variable length string */
         5 VSCCSID fixed bin(31);    /* CCSID of variable length string */
      3 OpenOptions fixed bin(31),   /* Failing open options */
      3 SubOptions fixed bin(31);    /* Failing subscription options */
  /* Ver:2 */
```

Deklaracja High Level Assembler dla usługi MQSTS

```
    MQSTS                         DSECT
    MQSTS_STRUCID                 DS    CL4   Structure identifier
    MQSTS_VERSION                 DS    F     Structure version number
    MQSTS_COMPCODE                DS    F     Completion code
    MQSTS_REASON                  DS    F     Reason code
    MQSTS_PUTSUCCESSCOUNT         DS    F     Success count
    MQSTS_PUTWARNINGCOUNT         DS    F     Warning count
    MQSTS_PUTFAILURECOUNT         DS    F     Failure count
    MQSTS_OBJTYPE                 DS    F     Object type
    MQSTS_OBJNAME                 DS    CL48  Object name
    MQSTS_OBJQMGR                 DS    CL48  Object queue manager
```

```
MQSTS_ROBJNAME                DS    CL48  Resolved object name
MQSTS_ROBJQMGR                DS    CL48  Resolved object queue manager
MQSTS_OBJECTSTRING            DS    0F   Force fullword alignment
MQSTS_OBJECTSTRING_VSPTR      DS    A  Address of variable length string
MQSTS_OBJECTSTRING_VSOFFSET   DS    F  Offset of variable length string
MQSTS_OBJECTSTRING_VSBUFSIZE  DS    F  Size of buffer
MQSTS_OBJECTSTRING_VSLENGTH   DS    F  Length of variable length string
MQSTS_OBJECTSTRING_VSCCSID    DS    F  CCSID of variable length string
MQSTS_OBJECTSTRING_LENGTH     EQU   *-MQSTS_OBJECTSTRING
                                    ORG   MQSTS_OBJECTSTRING
MQSTS_OBJECTSTRING_AREA       DS    CL(MQSTS_OBJECTSTRING_LENGTH)
*
MQSTS_SUBNAME                 DS    0F Force fullword alignment
MQSTS_SUBNAME_VSPTR           DS    A  Address of variable length string
MQSTS_SUBNAME_VSOFFSET        DS    F  Offset of variable length string
MQSTS_SUBNAME_VSBUFSIZE       DS    F  Size of buffer
MQSTS_SUBNAME_VSLENGTH        DS    F  Length of variable length string
MQSTS_SUBNAME_VSCCSID         DS    F  CCSID of variable length string
MQSTS_SUBNAME_LENGTH          EQ    *-MQSTS_SUBNAME
                                    ORG   MQSTS_SUBNAME
MQSTS_SUBNAME_AREA            DS    CL(MQSTS_SUBNAME_LENGTH)
*
MQSTS_OPENOPTIONS             DS    F  Failing open options
MQSTS_SUBOPTIONS              DS    F  Failing subscription option
MQSTS_LENGTH                  EQU   *-MQSTS
                              ORG   MQSTS
MQSTS_AREA                    DS    CL(MQSTS_LENGTH)
```

Odsyłacze pokrewne

“MQSTAT-pobieranie informacji o statusie” na stronie 809

Użyj wywołania MQSTAT, aby pobrać informacje o statusie. Typ zwracanych informacji o statusie jest określany przez wartość typu określoną w wywołaniu.

StrucId (MQCHAR4) dla MQSTS

Jest to identyfikator struktury struktury raportowania statusu. Jest to zawsze pole wejściowe. Jego wartością jest MQSTS_STRUC_ID.

Wartość musi być następująca:

MQSTS_STRUC_ID (ID struktury MQSTS)
Identyfikator struktury raportowania statusu.

Dla języka programowania C zdefiniowana jest również stała MQSTS_STRUC_ID_ARRAY . Ma taką samą wartość jak MQSTS_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Wersja (MQLONG) dla MQSTS

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

MQSTS_VERSION_1
Struktura raportowania statusu w wersji 1.

MQSTS_VERSION_2
Struktura raportowania statusu wersji 2.

Następująca stała określa numer wersji bieżącej:

BIEŻĄCA_WERSJA_MQSTS
Bieżąca wersja struktury raportowania statusu. Bieżąca wersja to MQSTS_VERSION_2.

Version jest zawsze zmienną wejściową. Jego wartością początkową jest MQSTS_VERSION_1.

CompCode (MQLONG) dla usługi MQSTS

Kod zakończenia raportowanej operacji.

Interpretacja parametru CompCode zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Jest to kod zakończenia wynikający z poprzedniej asynchronicznej operacji umieszczania na obiekcie określonym w parametrze ObjectName.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Jeśli połączenie jest ponownie nawiązywane lub ponowne nawiązywanie połączenia nie powiodło się, jest to kod zakończenia, który spowodował rozpoczęcie ponownego nawiązywania połączenia.

Jeśli połączenie jest obecnie połączone, wartością jest MQCC_OK.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli ponowne połączenie nie powiodło się, jest to kod zakończenia, który spowodował niepowodzenie ponownego połączenia.

Jeśli połączenie jest obecnie połączone lub nawiązywane ponownie, wartością jest MQCC_OK.

CompCode jest zawsze zmienną wyjściową. Jego wartością początkową jest MQCC_OK.

Przyczyna (MQLONG) dla MQSTS

Kod przyczyny raportowanej operacji.

Interpretacja parametru Reason zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Jest to kod przyczyny wynikający z poprzedniej asynchronicznej operacji umieszczania (put) dla obiektu określonego w parametrze ObjectName.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Jeśli połączenie jest ponownie nawiązywane lub ponowne nawiązywanie połączenia nie powiodło się, jest to kod przyczyny, który spowodował rozpoczęcie ponownego nawiązywania połączenia.

Jeśli połączenie jest obecnie połączone, wartością jest MQRC_NONE.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli ponowne nawiązanie połączenia nie powiodło się, jest to kod przyczyny, który spowodował niepowodzenie ponownego nawiązania połączenia.

Jeśli połączenie jest obecnie połączone lub nawiązywane ponownie, wartością jest MQRC_NONE.

Reason jest zmienną wyjściową. Jego wartością początkową jest MQRC_NONE.

PutSuccess(MQLONG) dla MQSTS

Liczba zakończonych powodzeniem asynchronicznych operacji umieszczania.

Wartość parametru PutSuccessCount zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Liczba asynchronicznych operacji put dla obiektu o nazwie w strukturze MQSTS , które zakończyły się MQCC_OK.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Zero.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Zero.

PutSuccessCount jest zmienną wyjściową. Wartością początkową jest zero.

PutWarning(MQLONG) dla MQSTS

Liczba asynchronicznych operacji umieszczania, które zakończyły się z ostrzeżeniem.

Wartość parametru PutWarningCount zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Liczba asynchronicznych operacji put dla obiektu o nazwie w strukturze MQSTS , które zakończyły się MQCC_WARNING.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Zero.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Zero.

PutWarningCount jest zmienną wyjściową. Wartością początkową jest zero.

Liczba wywołań PutFailure(MQLONG) dla usługi MQSTS

Liczba asynchronicznych operacji umieszczania, które się nie powiodły.

Wartość parametru PutFailureCount zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Liczba asynchronicznych operacji put dla obiektu o nazwie w strukturze MQSTS , które zakończyły się MQCC_FAILED.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Zero.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Zero.

PutFailureCount jest zmienną wyjściową. Wartością początkową jest zero.

ObjectType (MQLONG) dla usługi MQSTS

Typ obiektu, którego nazwa jest zgłaszana w programie *ObjectName* .

Możliwe wartości parametru ObjectType zawiera “MQOT_ * (typy obiektów i rozszerzone typy obiektów)” na stronie 165.

ObjectType jest zmienną wyjściową. Jego wartością początkową jest MQOT_Q.

ObjectName (MQCHAR48) dla usługi MQSTS

Nazwa zgłaszanego obiektu.

Interpretacja parametru ObjectName zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Jest to nazwa kolejki lub tematu używanego w operacji put, której niepowodzenie jest zgłaszane w polach *CompCode* i *Reason* w strukturze MQSTS .

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Jeśli połączenie jest nawiązywane ponownie, jest to nazwa menedżera kolejek powiązanego z połączeniem.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli ponowne nawiązanie połączenia nie powiodło się, jest to nazwa obiektu, który spowodował niepowodzenie ponownego nawiązania połączenia. Przyczyna niepowodzenia jest zgłaszana w polach *CompCode* i *Reason* w strukturze MQSTS .

ObjectName jest zmienną wyjściową. Jego wartością początkową jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ObjectQMgrNazwa (MQCHAR48) dla MQSTS

Nazwa zgłaszanego menedżera kolejek.

Interpretacja parametru `ObjectQMgrName` zależy od wartości parametru `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Jest to nazwa menedżera kolejek, w którym zdefiniowano obiekt *ObjectName* . Nazwa, która jest całkowicie pusta aż do pierwszego znaku o kodzie zero lub końca pola, oznacza menedżera kolejek, z którym połączona jest aplikacja (menedżer kolejek lokalnych).

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Multi

Pole **ObjectQMgrName** zawiera nazwę menedżera kolejek, z którym żądane jest ponowne nawiązanie połączenia, lub jest puste, jeśli nie określono żadnego menedżera kolejek. Jeśli to możliwe, klient próbuje ponownie nawiązać połączenie z menedżerem kolejek o tej nazwie.

z/OS

Puste.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Jeśli ponowne nawiązanie połączenia nie powiodło się, jest to nazwa obiektu, który spowodował niepowodzenie ponownego nawiązania połączenia. Przyczyna niepowodzenia jest zgłaszana w polach *CompCode* i *Reason* w strukturze `MQSTS` .

`ObjectQMgrName` jest zmienną wyjściową. Jego wartością jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ResolvedObjectNazwa (MQCHAR48) obiektu MQSTS

Nazwa obiektu nazwanego w programie *ObjectName* po przetłumaczeniu nazwy przez lokalny menedżer kolejek.

Interpretacja parametru `ResolvedObjectName` zależy od wartości parametru `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedObjectName` jest nazwą obiektu, którego nazwa została podana w pliku *ObjectName* po przetłumaczeniu nazwy przez lokalny menedżer kolejek. Zwracana nazwa jest nazwą obiektu istniejącego w menedżerze kolejek identyfikowanym przez *ResolvedQMgrName*.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Puste.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Puste.

`ResolvedObjectName` jest zmienną wyjściową. Jego wartością początkową jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ResolvedQMgrNazwa (MQCHAR48) dla usługi MQSTS

Nazwa docelowego menedżera kolejek po przetłumaczeniu nazwy przez lokalny menedżer kolejek.

Interpretacja parametru `ResolvedQMgrName` zależy od wartości parametru `MQSTAT` **Type** .

MQSTAT_TYPE_ASYNC_ERROR

`ResolvedQMgrName` jest nazwą docelowego menedżera kolejek po przetłumaczeniu nazwy przez lokalny menedżer kolejek. Zwracana nazwa to nazwa menedżera kolejek, który jest właścicielem obiektu identyfikowanego przez *ResolvedObjectName*. *ResolvedQMgrName* może być nazwą lokalnego menedżera kolejek.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Puste.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Puste.

ResolvedQMgrName jest zawsze zmienną wyjściową. Jego wartością początkową jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ObjectString (MQCHARV) dla MQSTS

Długa nazwa obiektu, dla którego zgłoszono błąd. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

Interpretacja parametru ObjectString zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Jest to długa nazwa obiektu kolejki lub tematu używanego w operacji MQPUT , która się nie powiodła.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Łańcuch o zerowej długości

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Jest to długa nazwa obiektu, który spowodował niepowodzenie ponownego połączenia.

ObjectString jest zmienną wyjściową. Jego wartością początkową jest łańcuch o zerowej długości.

SubName (MQCHARV) dla MQSTS

Nazwa subskrypcji zakończonej niepowodzeniem. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

Interpretacja parametru SubName zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Łańcuch o zerowej długości.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Łańcuch o zerowej długości.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Nazwa subskrypcji, która spowodowała niepowodzenie ponownego połączenia. Jeśli nazwa subskrypcji nie jest dostępna lub niepowodzenie nie jest związane z subskrypcją, jest to łańcuch o zerowej długości.

SubName jest zmienną wyjściową. Jego wartością początkową jest łańcuch o zerowej długości.

OpenOptions (MQLONG) dla MQSTS

OpenOptions używany do otwierania zgłaszanego obiektu. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

Wartość parametru OpenOptions zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Zero.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Zero.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Wartość OpenOptions używana w momencie wystąpienia awarii. Przyczyna niepowodzenia jest zgłaszana w polach *CompCode* i *Reason* w strukturze MQSTS .

OpenOptions jest zmienną wyjściową. Wartością początkową jest zero.

SubOptions (MQLONG) dla usługi MQSTS

SubOptions używany do otwierania subskrypcji, która się nie powiodła. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

Interpretacja parametru SubOptions zależy od wartości parametru MQSTAT **Type** .

MQSTAT_TYPE_ASYNC_ERROR

Zero.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Zero.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Wartość SubOptions używana w momencie wystąpienia awarii. Jeśli niepowodzenie nie jest związane z subskrybowaniem tematu, zwracana jest wartość zero.

SubOptions jest zmienną wyjściową. Wartością początkową jest zero.

MQTM-komunikat wyzwalacza

Struktura MQTM opisuje dane w komunikacie wyzwalacza, który jest wysyłany przez menedżer kolejek do aplikacji monitorującej wyzwalacz, gdy dla kolejki wystąpi zdarzenie wyzwalacza.Ta struktura jest częścią interfejsu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska IBM MQ .

Nazwa formatu

MQFMT_TRIGGER.

Zestaw znaków i kodowanie

Dane znakowe w produkcie MQTM znajdują się w zestawie znaków menedżera kolejek, który generuje produkt MQTM. Dane liczbowe w produkcie MQTM są kodowane na komputerze menedżera kolejek, który generuje produkt MQTM.

Zestaw znaków i kodowanie produktu MQTM są określone przez pola *CodedCharSetId* i *Encoding* w następujących polach:

- MQMD (jeśli struktura MQTM znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQTM (wszystkie inne przypadki).

Użycie

Aplikacja monitorująca wyzwalacz może wymagać przekazania niektórych lub wszystkich informacji zawartych w komunikacie wyzwalacza do aplikacji uruchamianej przez aplikację monitorującą wyzwalacz. Informacje, które mogą być wymagane przez uruchomioną aplikację, obejmują *QName*, *TriggerData*i *UserData*. Aplikacja monitora wyzwalacza może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2 , w zależności od tego, co jest dozwolone w środowisku i wygodne dla uruchomionej aplikacji. Więcej informacji na temat komendy MQTMC2zawiera sekcja “MQTMC2 -komunikat wyzwalacza 2 (format znakowy)” na stronie 625.

- z/OS W systemie z/OSdla aplikacji MQAT_CICS, która jest uruchamiana przy użyciu transakcji CKTI, cała struktura komunikatu wyzwalacza MQTM jest udostępniana dla uruchomionej transakcji. Informacje można pobrać za pomocą komendy EXEC CICS RETRIEVE.
- IBM i W systemie IBM iaplikacja monitora wyzwalacza dostarczana z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.

Informacje na temat używania wyzwalaczy zawiera sekcja Uruchamianie aplikacji IBM MQ za pomocą wyzwalaczy.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 533. Pola w MQTM dla MQTM

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQTM_STRUC_ID	'TM¬¬'
Wersja (numer wersji struktury)	MQTM_VERSION_1	1
QName (nazwa wyzwalanej kolejki)	Brak	Pusty łańcuch lub odstępy
ProcessName (nazwa obiektu procesu)	Brak	Pusty łańcuch lub odstępy
TriggerData (dane wyzwalacza)	Brak	Pusty łańcuch lub odstępy
ApplType (typ aplikacji)	Brak	0
ApplId (identyfikator aplikacji)	Brak	Pusty łańcuch lub odstępy
EnvData (dane środowiska)	Brak	Pusty łańcuch lub odstępy
UserData (dane użytkownika)	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQTM_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQTM MyTM = {MQTM_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQTM

```
typedef struct tagMQTM MQTM;
struct tagMQTM {
  MQCHAR4    StrucId;      /* Structure identifier */
  MQLONG     Version;      /* Structure version number */
  MQCHAR48   QName;        /* Name of triggered queue */
  MQCHAR48   ProcessName;  /* Name of process object */
  MQCHAR64   TriggerData;  /* Trigger data */
  MQLONG     ApplType;     /* Application type */
  MQCHAR256  ApplId;       /* Application identifier */
  MQCHAR128  EnvData;      /* Environment data */
  MQCHAR128  UserData;     /* User data */
};
```

Deklaracja języka COBOL dla produktu MQTM

```
**   MQTM structure
  10 MQTM.
**    Structure identifier
   15 MQTM-STRUCID     PIC X(4).
**    Structure version number
   15 MQTM-VERSION     PIC S9(9) BINARY.
**    Name of triggered queue
   15 MQTM-QNAME       PIC X(48).
**    Name of process object
   15 MQTM-PROCESSNAME PIC X(48).
**    Trigger data
   15 MQTM-TRIGGERDATA PIC X(64).
**    Application type
   15 MQTM-APPLTYPE    PIC S9(9) BINARY.
**    Application identifier
   15 MQTM-APPLID      PIC X(256).
**    Environment data
   15 MQTM-ENVDATA     PIC X(128).
**    User data
   15 MQTM-USERDATA    PIC X(128).
```

Deklaracja PL/I dla MQTM

```
dcl
 1 MQTM based,
   3 StrucId     char(4),       /* Structure identifier */
   3 Version     fixed bin(31), /* Structure version number */
   3 QName       char(48),      /* Name of triggered queue */
   3 ProcessName char(48),      /* Name of process object */
   3 TriggerData char(64),      /* Trigger data */
   3 ApplType    fixed bin(31), /* Application type */
   3 ApplId      char(256),     /* Application identifier */
   3 EnvData     char(128),     /* Environment data */
   3 UserData    char(128);     /* User data */
```

Deklaracja High Level Assembler dla MQTM

```
MQTM              DSECT
MQTM_STRUCID      DS   CL4    Structure identifier
MQTM_VERSION      DS   F      Structure version number
MQTM_QNAME        DS   CL48   Name of triggered queue
MQTM_PROCESSNAME  DS   CL48   Name of process object
MQTM_TRIGGERDATA  DS   CL64   Trigger data
MQTM_APPLTYPE     DS   F      Application type
MQTM_APPLID       DS   CL256  Application identifier
MQTM_ENVDATA      DS   CL128  Environment data
MQTM_USERDATA     DS   CL128  User data
*
MQTM_LENGTH       EQU  *-MQTM
                  ORG  MQTM
MQTM_AREA         DS   CL(MQTM_LENGTH)
```

Deklaracja Visual Basic dla MQTM

```
Type MQTM
  StrucId     As String*4   'Structure identifier'
  Version     As Long       'Structure version number'
  QName       As String*48  'Name of triggered queue'
  ProcessName As String*48  'Name of process object'
  TriggerData As String*64  'Trigger data'
  ApplType    As Long       'Application type'
  ApplId      As String*256 'Application identifier'
  EnvData     As String*128 'Environment data'
  UserData    As String*128 'User data'
End Type
```

MQMD dla komunikatu wyzwalacza

Tabela 534. Ustawienia pól w strukturze MQMD komunikatu wyzwalacza wygenerowanego przez menedżer kolejek

Pole w strukturze MQMD	Użyta wartość
StrucId	MQMD_STRUC_ID (Identyfikator struktury kolejki)
Version	MQMD_VERSION_1
Report	MQRO_BRAK
MsgType	MQMT_DATAGRAM,
Expiry	MQEI_UNLIMITED,
Feedback	MQFB_BRAK
Encoding	RODZIMA MQENC
CodedCharSetId	Atrybut **CodedCharSetId** menedżera kolejek
Format	MQFMT_TRIGGER,
Priority	Atrybut **DefPriority** kolejki inicjującej
Persistence	MQPER_NOT_PERSISTENT
MsgId	Wartość unikalna
CorrelId	MQCI_BRAK
BackoutCount	0
ReplyToQ	Puste
ReplyToQMgr	Nazwa menedżera kolejek.
UserIdentifier	Puste
AccountingToken	MQACT_NONE
ApplIdentityData	Puste
PutApplType	MQAT_QMGR lub odpowiednio dla agenta kanału komunikatów
PutApplName	Pierwsze 28 bajtów nazwy menedżera kolejek
PutDate	Data wysłania komunikatu wyzwalacza
PutTime	Czas wysłania komunikatu wyzwalacza
ApplOriginData	Puste

Zaleca się, aby aplikacja generująca komunikat wyzwalacza ustawiła podobne wartości, z wyjątkiem następujących:

- Pole *Priority* można ustawić na wartość MQPRI_PRIORITY_AS_Q_DEF (menedżer kolejek zmieni ten priorytet na domyślny dla kolejki inicjującej po umieszczeniu komunikatu).
- Pole *ReplyToQMgr* można ustawić na wartość pustą (menedżer kolejek zmieni tę wartość na nazwę lokalnego menedżera kolejek podczas umieszczania komunikatu).
- Ustaw pola kontekstu odpowiednio dla aplikacji.

StrucId (MQCHAR4) dla MQTM

Jest to identyfikator struktury struktury komunikatu wyzwalacza. Jest to zawsze pole wejściowe. Jego wartością jest MQTM_STRUC_ID.

Wartość musi być następująca:

MQTM_STRUC_ID
Identyfikator struktury komunikatu wyzwalacza.

Dla języka programowania C zdefiniowana jest również stała MQTM_STRUC_ID_ARRAY. Ma taką samą wartość jak MQTM_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQTM

Jest to numer wersji struktury. Wartość musi być następująca:

MQTM_VERSION_1
Numer wersji struktury komunikatu wyzwalacza.

Następująca stała określa numer wersji bieżącej:

MQTM_CURRENT_VERSION
Bieżąca wersja struktury komunikatu wyzwalacza.

Wartością początkową tego pola jest MQTM_VERSION_1.

Nazwa QName (MQCHAR48) dla usługi MQTM

Jest to nazwa kolejki, dla której wystąpiło zdarzenie wyzwalające i jest używana przez aplikację uruchomioną przez aplikację monitorującą wyzwalacz. Menedżer kolejek inicjuje to pole wartością atrybutu **QName** wyzwalanej kolejki. Szczegółowe informacje na temat tego atrybutu zawiera sekcja“Atrybuty kolejek” na stronie 863.

Nazwy, które są krótsze niż zdefiniowana długość pola, są dopełniane do prawej strony odstępami; nie są przedwcześnie kończone znakiem o kodzie zero.

Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

ProcessName (MQCHAR48) dla MQTM

Jest to nazwa obiektu procesu menedżera kolejek określona dla wyzwalanej kolejki i może być używana przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **ProcessName** kolejki identyfikowanej przez pole *QName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja“Atrybuty kolejek” na stronie 863.

Nazwy krótsze niż zdefiniowana długość pola są zawsze dopełniane do prawej strony odstępami; nie są przedwcześnie kończone znakiem o kodzie zero.

Długość tego pola jest określona przez wartość MQ_PROCESS_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

TriggerData (MQCHAR64) dla MQTM

Są to dane w formacie swobodnym używane przez aplikację monitora wyzwalacza, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **TriggerData** kolejki identyfikowanej przez pole *QName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty kolejek” na stronie 863 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

W systemie z/OSinformacje te nie są używane w przypadku aplikacji CICS uruchamianej przy użyciu transakcji CKTI.

Długość tego pola jest określona przez parametr MQ_TRIGGER_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 64 znaki puste w innych językach programowania.

ApplType (MQLONG) dla MQTM

Identyfikuje on rodzaj programu, który ma zostać uruchomiony i jest używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **ApplType** obiektu procesu identyfikowanego przez pole *ProcessName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów” na stronie 900 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

ApplType może mieć jedną z następujących wartości standardowych. Typy zdefiniowane przez użytkownika mogą być również używane, ale powinny być ograniczone do wartości z zakresu od MQAT_USER_FIRST do MQAT_USER_LAST:

MQAT_AIX
Aplikacja AIX (taka sama jak MQAT_UNIX).

MQAT_BATCH
aplikacja wsadowa

MQAT_BROKER
Aplikacja brokera

MQAT_CICS
CICS .

MQAT_CICS_BRIDGE
Aplikacja CICS bridge .

MQAT_CICS_VSE
CICS/VSE .

MQAT_DOS
Aplikacja IBM MQ MQI client w systemie DOS na komputerze PC.

MQAT_IMS
Aplikacja IMS .

MQAT_IMS_BRIDGE,
Aplikacja mostu IMS .

MQAT_JAVA
Aplikacja Java .

MQAT_MVS
Aplikacja MVS lub TSO (taka sama wartość jak MQAT_ZOS).

MQAT_NOTES_AGENT
Lotus Notes Aplikacja agenta.

MQAT_OS390
Aplikacja OS/390 (taka sama jak MQAT_ZOS).

MQAT_OS400
Aplikacja IBM i .

MQAT_RRS_BATCH,
Aplikacja wsadowa RRS.

MQAT_UNIX
Aplikacja UNIX .

MQAT_UNKNOWN
Zastosowanie nieznanego typu.

UŻYTKOWNIK_MQAT
Typ aplikacji zdefiniowany przez użytkownika.

MQAT_VOS
Aplikacja Stratus VOS.

MQAT_WINDOWS
16-bitowa aplikacja Windows .

MQAT_WINDOWS_NT
32-bitowa aplikacja Windows .

MQAT_WLM
Aplikacja menedżera obciążenia z/OS .

MQAT_XCF,
XCF.

MQAT_ZOS
Aplikacja z/OS .

MQAT_USER_FIRST
Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Tabela MQAT_USER_LAST
Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Wartością początkową tego pola jest `0`.

ApplId (MQCHAR256) dla MQTM

Jest to łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona i używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **ApplId** obiektu procesu identyfikowanego przez pole *ProcessName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów” na stronie 900 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

Znaczenie parametru *ApplId* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniony przez IBM MQ wymaga, aby *ApplId* była nazwą programu wykonywalnego. Poniższe uwagi dotyczą wskazanych środowisk:

- W systemie z/OS *ApplId* jest to:
  - Identyfikator transakcji CICS dla aplikacji uruchomionych za pomocą transakcji CKTI monitora wyzwalacza CICS
  - Identyfikator transakcji IMS dla aplikacji uruchomionych za pomocą monitora wyzwalacza IMS CSQQTRMN
- W systemach Windows nazwa programu może być poprzedzona ścieżką do napędu i katalogu.
- W systemie IBM inazwa programu może być poprzedzona nazwą biblioteki i znakiem/.
- W systemie AIX and Linuxnazwa programu może być poprzedzona ścieżką do katalogu.

Długość tego pola jest określona przez wartość MQ_PROCESS_APPL_ID_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 256 znaków odstępu w innych językach programowania.

EnvData (MQCHAR128) dla MQTM

Jest to łańcuch znaków, który zawiera informacje związane ze środowiskiem dotyczące aplikacji, która ma zostać uruchomiona, i jest używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **EnvData** obiektu procesu identyfikowanego przez pole *ProcessName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów” na stronie 900 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

W systemie z/OSinformacje te nie są używane w przypadku aplikacji CICS uruchamianej za pomocą transakcji CKTI lub aplikacji IMS uruchamianej za pomocą transakcji CSQQTRMN.

Długość tego pola jest określona przez wartość MQ_PROCESS_ENV_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 128 znaków odstępu w innych językach programowania.

UserData (MQCHAR128) dla MQTM

Jest to łańcuch znaków, który zawiera informacje o użytkowniku dotyczące uruchamianej aplikacji i jest używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **UserData** obiektu procesu identyfikowanego przez pole *ProcessName* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów” na stronie 900 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

W przypadku systemu Microsoft Windowsłańcuch znaków nie może zawierać podwójnych cudzysłowów, jeśli definicja procesu ma zostać przekazana do programu **runmqtrm**.

Długość tego pola jest określona przez wartość MQ_PROCESS_USER_DATA_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 128 znaków odstępu w innych językach programowania.

MQTMC2 -komunikat wyzwalacza 2 (format znakowy)

Gdy aplikacja monitora wyzwalacza pobiera komunikat wyzwalacza (MQTM) z kolejki inicjującej, może być konieczne przekazanie przez monitor wyzwalacza niektórych lub wszystkich informacji zawartych w komunikacie wyzwalacza do aplikacji uruchamianej przez monitor wyzwalacza.

Informacje, które mogą być potrzebne uruchomionej aplikacji, obejmują *QName*, *TriggerData*i *UserData*. Aplikacja monitora wyzwalacza może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2 , w zależności od tego, co jest dozwolone w środowisku i wygodne dla uruchomionej aplikacji.

Ta struktura jest częścią interfejsu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska IBM MQ .

Zestaw znaków i kodowanie

Dane znakowe w programie MQTMC2 znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek systemu **CodedCharSetId** .

Użycie

Struktura MQTMC2 jest bardzo podobna do formatu struktury MQTM. Różnica polega na tym, że pola nieznakowe w programie MQTM są zmieniane w programie MQTMC2 na pola znakowe o tej samej długości, a nazwa menedżera kolejek jest dodawana na końcu struktury.

- z/OS W systemie z/OSw przypadku aplikacji MQAT_IMS uruchamianej za pomocą aplikacji CSQQTRMN struktura MQTMC2 jest udostępniana uruchomionej aplikacji.
- IBM i W systemie IBM iaplikacja monitora wyzwalacza dostarczana z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 535. Pola w tabeli MQTMC2

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQTMC_STRUC_ID (ID struktury MQTMC)	'TMC¬'
Wersja (numer wersji struktury)	MQTMC_VERSION_2	'¬¬¬2'
QName (nazwa wyzwalanej kolejki)	Brak	Pusty łańcuch lub odstępy
ProcessName (nazwa obiektu procesu)	Brak	Pusty łańcuch lub odstępy
TriggerData (dane wyzwalacza)	Brak	Pusty łańcuch lub odstępy
ApplType (typ aplikacji)	Brak	Puste

Tabela 535. Pola w tabeli MQTMC2 (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
ApplId (identyfikator aplikacji)	Brak	Pusty łańcuch lub odstępy
EnvData (dane środowiska)	Brak	Pusty łańcuch lub odstępy
UserData (dane użytkownika)	Brak	Pusty łańcuch lub odstępy
QMgrName (nazwa menedżera kolejek)	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQTMC2_DEFAULT zawiera wartości wymienione powyżej. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQTMC2 MyTMC = {MQTMC2_DEFAULT};
```

Deklaracje językowe

Deklaracja-C dla MQTMC2

```
typedef struct tagMQTMC2 MQTMC2;
struct tagMQTMC2 {
  MQCHAR4    StrucId;      /* Structure identifier */
  MQCHAR4    Version;      /* Structure version number */
  MQCHAR48   QName;        /* Name of triggered queue */
  MQCHAR48   ProcessName;  /* Name of process object */
  MQCHAR64   TriggerData;  /* Trigger data */
  MQCHAR4    ApplType;     /* Application type */
  MQCHAR256  ApplId;       /* Application identifier */
  MQCHAR128  EnvData;      /* Environment data */
  MQCHAR128  UserData;     /* User data */
  MQCHAR48   QMgrName;     /* Queue manager name */
};
```

Deklaracja języka COBOL dla MQTMC2

```
**   MQTMC2 structure
  10 MQTMC2.
**    Structure identifier
   15 MQTMC2-STRUCID     PIC X(4).
**    Structure version number
   15 MQTMC2-VERSION     PIC X(4).
**    Name of triggered queue
   15 MQTMC2-QNAME       PIC X(48).
**    Name of process object
   15 MQTMC2-PROCESSNAME PIC X(48).
**    Trigger data
   15 MQTMC2-TRIGGERDATA PIC X(64).
**    Application type
   15 MQTMC2-APPLTYPE    PIC X(4).
**    Application identifier
   15 MQTMC2-APPLID      PIC X(256).
**    Environment data
   15 MQTMC2-ENVDATA     PIC X(128).
**    User data
```

```
   15 MQTMC2-USERDATA    PIC X(128).
**    Queue manager name
   15 MQTMC2-QMGRNAME    PIC X(48).
```

Deklaracja języka PL/I dla MQTMC2

```
dcl
 1 MQTMC2 based,
  3 StrucId     char(4),   /* Structure identifier */
  3 Version     char(4),   /* Structure version number */
  3 QName       char(48),  /* Name of triggered queue */
  3 ProcessName char(48),  /* Name of process object */
  3 TriggerData char(64),  /* Trigger data */
  3 ApplType    char(4),   /* Application type */
  3 ApplId      char(256), /* Application identifier */
  3 EnvData     char(128), /* Environment data */
  3 UserData    char(128), /* User data */
  3 QMgrName    char(48);  /* Queue manager name */
```

Deklaracja High Level Assembler dla MQTMC2

```
MQTMC2              DSECT
MQTMC2_STRUCID      DS   CL4    Structure identifier
MQTMC2_VERSION      DS   CL4    Structure version number
MQTMC2_QNAME        DS   CL48   Name of triggered queue
MQTMC2_PROCESSNAME  DS   CL48   Name of process object
MQTMC2_TRIGGERDATA  DS   CL64   Trigger data
MQTMC2_APPLTYPE     DS   CL4    Application type
MQTMC2_APPLID       DS   CL256  Application identifier
MQTMC2_ENVDATA      DS   CL128  Environment data
MQTMC2_USERDATA     DS   CL128  User data
MQTMC2_QMGRNAME     DS   CL48   Queue manager name
*
MQTMC2_LENGTH       EQU  *-MQTMC2
                    ORG  MQTMC2
MQTMC2_AREA         DS   CL(MQTMC2_LENGTH)
```

Deklaracja Visual Basic dla MQTMC2

```
Type MQTMC2
  StrucId     As String*4   'Structure identifier'
  Version     As String*4   'Structure version number'
  QName       As String*48  'Name of triggered queue'
  ProcessName As String*48  'Name of process object'
  TriggerData As String*64  'Trigger data'
  ApplType    As String*4   'Application type'
  ApplId      As String*256 'Application identifier'
  EnvData     As String*128 'Environment data'
  UserData    As String*128 'User data'
  QMgrName    As String*48  'Queue manager name'
End Type
```

StrucId (MQCHAR4) dla MQTMC2

Jest to identyfikator struktury struktury komunikatu wyzwalacza 2 (format znakowy). Jest to zawsze pole wejściowe. Jego wartością jest MQTMC2_STRUC_ID.

Wartość musi być następująca:

MQTMC2_STRUC_ID
Identyfikator struktury komunikatu wyzwalacza (format znakowy).

Dla języka programowania C zdefiniowana jest również stała MQTMC2_STRUC_ID_ARRAY . Ma taką samą wartość jak MQTMC2_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQCHAR4) dla MQTMC2

Numer wersji struktury.

Wartość musi być następująca:

MQTMC_VERSION_2
Struktura komunikatu wyzwalacza (format znakowy) w wersji 2.

Dla języka programowania C zdefiniowana jest również stała MQTMC_VERSION_2_ARRAY ; ma ona taką samą wartość jak MQTMC_VERSION_2, ale jest tablicą znaków, a nie łańcuchem.

Następująca stała określa numer wersji bieżącej:

MQTMC_CURRENT_VERSION
Bieżąca wersja struktury komunikatu wyzwalacza (format znakowy).

QName (MQCHAR48) dla MQTMC2
Nazwa wyzwalanej kolejki.

Patrz pole *QName* w strukturze MQTM.

ProcessName (MQCHAR48) dla MQTMC2
Nazwa obiektu procesu.

Patrz pole *ProcessName* w strukturze MQTM.

TriggerData (MQCHAR64) dla MQTMC2
Dane wyzwalacza.

Patrz pole *TriggerData* w strukturze MQTM.

ApplType (MQCHAR4) dla MQTMC2
Typ aplikacji.

To pole zawsze zawiera odstępy, niezależnie od wartości w polu *ApplType* w strukturze MQTM oryginalnego komunikatu wyzwalacza.

ApplId (MQCHAR256) dla MQTMC2

Identyfikator aplikacji.

Patrz pole *ApplId* w strukturze MQTM.

EnvData (MQCHAR128) dla MQTMC2
Dane środowiska.

Patrz pole *EnvData* w strukturze MQTM.

UserData (MQCHAR128) dla MQTMC2
Dane użytkownika.

Patrz pole *UserData* w strukturze MQTM.

QMgrName (MQCHAR48) dla MQTMC2
Nazwa menedżera kolejek.

Jest to nazwa menedżera kolejek, w którym wystąpiło zdarzenie wyzwalające.

MQWIH-nagłówek informacji o pracy

Jeśli komunikat ma być przetwarzany przez menedżer obciążenia z/OS (WLM), musi zaczynać się od struktury MQWIH. Ta struktura opisuje informacje, które muszą być obecne na początku komunikatu, który ma być obsługiwany przez WLM.

Dostępność

Wszystkie systemy IBM MQ oraz klienty IBM MQ połączone z tymi systemami.

Nazwa formatu

MQFMT_WORK_INFO_HEADER.

Zestaw znaków i kodowanie

Pola w strukturze MQWIH mają zestaw znaków i kodowanie określone w polach *CodedCharSetId* i *Encoding* w strukturze nagłówka poprzedzającej MQWIH lub w tych polach w strukturze MQMD, jeśli MQWIH jest na początku danych komunikatu aplikacji.

Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Użycie

W przypadku dowolnej platformy obsługiwanej przez produkt IBM MQ można utworzyć i przesłać komunikat zawierający strukturę MQWIH, ale tylko menedżer kolejek systemu IBM MQ for z/OS może współpracować z produktem WLM. Dlatego aby komunikat mógł zostać odebrany do menedżera WLM z menedżera kolejek innego niżz/OS , sieć menedżera kolejek musi zawierać co najmniej jeden menedżer kolejek systemu z/OS , przez który może być kierowany komunikat.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 536. Pola w MQWIH

Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**
StrucId (identyfikator struktury)	MQWIH_ID_struktury	'WIH¬'
Wersja (numer wersji struktury)	MQWIH_VERSION_1	1
StrucLength (długość struktury MQWIH)	MQWIH_LENGTH_1	120
Kodowanie (kodowanie liczbowe danych następujących po MQWIH)	Brak	0
CodedCharSetId (identyfikator zestawu znaków danych następujący po MQWIH)	MQCCSI_XX_ENCODE_CASE_ONE niezdefiniowany	0
Format (nazwa formatu danych następująca po MQWIH)	MQFMT_BRAK	Puste
Flagi (flags)	MQWIH_BRAK	0
ServiceName (nazwa usługi)	Brak	Puste
ServiceStep (nazwa kroku usługi)	Brak	Puste
MsgToken (znacznik komunikatu)	MQMTOK_BRAK	Wartości null
Zarezerwowane (zastrzeżone)	Brak	Puste

Tabela 536. Pola w MQWIH (kontynuacja)		
Nazwa i opis pola	**Nazwa stałej**	**Wartość początkowa (jeśli istnieje) stałej**

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C: zmienna makraMQWIH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQWIH MyWIH = {MQWIH_DEFAULT};
```

Deklaracje językowe

Deklaracja C dla MQWIH

```
typedef struct tagMQWIH MQWIH;
struct tagMQWIH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQLONG    StrucLength;     /* Length of MQWIH structure */
  MQLONG    Encoding;        /* Numeric encoding of data that follows
                                MQWIH */
  MQLONG    CodedCharSetId;  /* Character-set identifier of data that
                                follows MQWIH */
  MQCHAR8   Format;          /* Format name of data that follows
                                MQWIH */
  MQLONG    Flags;           /* Flags */
  MQCHAR32  ServiceName;     /* Service name */
  MQCHAR8   ServiceStep;     /* Service step name */
  MQBYTE16  MsgToken;        /* Message token */
  MQCHAR32  Reserved;        /* Reserved */
};
```

Deklaracja języka COBOL dla MQWIH

```
**   MQWIH structure
  10 MQWIH.
**    Structure identifier
   15 MQWIH-STRUCID        PIC X(4).
**    Structure version number
   15 MQWIH-VERSION        PIC S9(9) BINARY.
**    Length of MQWIH structure
   15 MQWIH-STRUCLENGTH    PIC S9(9) BINARY.
**    Numeric encoding of data that follows MQWIH
   15 MQWIH-ENCODING       PIC S9(9) BINARY.
**    Character-set identifier of data that follows MQWIH
   15 MQWIH-CODEDCHARSETID PIC S9(9) BINARY.
**    Format name of data that follows MQWIH
   15 MQWIH-FORMAT         PIC X(8).
**    Flags
   15 MQWIH-FLAGS          PIC S9(9) BINARY.
**    Service name
   15 MQWIH-SERVICENAME    PIC X(32).
**    Service step name
   15 MQWIH-SERVICESTEP    PIC X(8).
**    Message token
   15 MQWIH-MSGTOKEN       PIC X(16).
**    Reserved
   15 MQWIH-RESERVED       PIC X(32).
```

Deklaracja języka PL/I dla MQWIH

```
dcl
 1 MQWIH based,
  3 StrucId        char(4),       /* Structure identifier */
  3 Version        fixed bin(31), /* Structure version number */
```

```
  3 StrucLength    fixed bin(31), /* Length of MQWIH structure */
  3 Encoding       fixed bin(31), /* Numeric encoding of data that
                                     follows MQWIH */
  3 CodedCharSetId fixed bin(31), /* Character-set identifier of data
                                     that follows MQWIH */
  3 Format         char(8),       /* Format name of data that follows
                                     MQWIH */
  3 Flags          fixed bin(31), /* Flags */
  3 ServiceName    char(32),      /* Service name */
  3 ServiceStep    char(8),       /* Service step name */
  3 MsgToken       char(16),      /* Message token */
  3 Reserved       char(32);      /* Reserved */
```

Deklaracja High Level Assembler dla MQWIH

```
MQWIH                 DSECT
MQWIH_STRUCID         DS   CL4   Structure identifier
MQWIH_VERSION         DS   F     Structure version number
MQWIH_STRUCLENGTH     DS   F     Length of MQWIH structure
MQWIH_ENCODING        DS   F     Numeric encoding of data that follows
*                                MQWIH
MQWIH_CODEDCHARSETID  DS   F     Character-set identifier of data that
*                                follows MQWIH
MQWIH_FORMAT          DS   CL8   Format name of data that follows MQWIH
MQWIH_FLAGS           DS   F     Flags
MQWIH_SERVICENAME     DS   CL32  Service name
MQWIH_SERVICESTEP     DS   CL8   Service step name
MQWIH_MSGTOKEN        DS   XL16  Message token
MQWIH_RESERVED        DS   CL32  Reserved
*
MQWIH_LENGTH          EQU  *-MQWIH
                      ORG  MQWIH
MQWIH_AREA            DS   CL(MQWIH_LENGTH)
```

Deklaracja Visual Basic dla MQWIH

```
Type MQWIH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  StrucLength    As Long      'Length of MQWIH structure'
  Encoding       As Long      'Numeric encoding of data that follows'
                              'MQWIH'
  CodedCharSetId As Long      'Character-set identifier of data that'
                              'follows MQWIH'
  Format         As String*8  'Format name of data that follows MQWIH'
  Flags          As Long      'Flags'
  ServiceName    As String*32 'Service name'
  ServiceStep    As String*8  'Service step name'
  MsgToken       As MQBYTE16  'Message token'
  Reserved       As String*32 'Reserved'
End Type
```

StrucId (MQCHAR4) dla MQWIH

Jest to identyfikator struktury struktury nagłówka informacji o pracy. Jest to zawsze pole wejściowe. Jego wartość to MQWIH_STRUC_ID.

Wartość musi być następująca:

MQWIH_ID_struktury
Identyfikator struktury nagłówka informacji o pracy.

Dla języka programowania C zdefiniowana jest również stała MQWIH_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQWIH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQWIH

Jest to numer wersji struktury. Wartość musi być następująca:

MQWIH_VERSION_1
Version-1 struktura nagłówka informacji o pracy.

Następująca stała określa numer wersji bieżącej:

MQWIH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka informacji o pracy.

Wartością początkową tego pola jest MQWIH_VERSION_1.

StrucLength (MQLONG) dla MQWIH

Jest to długość struktury MQWIH. Wartość musi być następująca:

MQWIH_LENGTH_1
Długość struktury nagłówka informacji o pracy w wersji version-1 .

Następująca stała określa długość bieżącej wersji:

MQWIH_CURRENT_LENGTH
Długość bieżącej wersji struktury nagłówka informacji o pracy.

Wartością początkową tego pola jest MQWIH_LENGTH_1.

Kodowanie (MQLONG) dla MQWIH

Określa kodowanie liczbowe danych, które są zgodne ze strukturą MQWIH. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

CodedCharSetId (MQLONG) dla MQWIH

Określa identyfikator zestawu znaków danych, które są zgodne ze strukturą MQWIH. Nie ma on zastosowania do danych znakowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

MQCCSI_INHERIT
Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość MQCCSI_INHERIT nie jest zwracana przez wywołanie MQGET.

Pola MQCCSI_INHERIT nie można używać, jeśli wartością pola *PutApplType* w deskryptorze MQMD jest MQAT_BROKER.

Wartością początkową tego pola jest MQCCSI_UNDEFINED.

Format (MQCHAR8) dla MQWIH

Określa nazwę formatu danych, które są zgodne ze strukturą MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *Format* w strukturze MQMD.

Długość tego pola jest określona przez wartość MQ_FORMAT_LENGTH. Wartością początkową tego pola jest MQFMT_NONE.

Flagi (MQLONG) dla MQWIH

Wartość musi być następująca:

MQWIH_BRAK
Brak flag.

Wartością początkową tego pola jest MQWIH_NONE.

ServiceName (MQCHAR32) dla MQWIH

Jest to nazwa usługi, która ma przetworzyć komunikat.

Długość tego pola jest określona przez wartość MQ_SERVICE_NAME_LENGTH. Wartość początkowa tego pola to 32 znaki odstępu.

ServiceStep (MQCHAR8) dla MQWIH

Jest to nazwa kroku *ServiceName* , do którego odnosi się komunikat.

Długość tego pola jest określona przez wartość MQ_SERVICE_STEP_LENGTH. Wartość początkowa tego pola to 8 znaków odstępu.

MsgToken (MQBYTE16) w środowisku MQWIH

Jest to znacznik komunikatu, który jednoznacznie identyfikuje komunikat.

W przypadku wywołań MQPUT i MQPUT1 to pole jest ignorowane. Długość tego pola jest określona przez wartość MQ_MSG_TOKEN_LENGTH. Wartością początkową tego pola jest MQMTOK_NONE.

Zarezerwowane (MQCHAR32) dla MQWIH

Jest to pole zastrzeżone; musi być puste.

MQXP-blok parametru wyjścia

Struktura MQXP jest używana jako parametr wejścia/wyjścia dla wyjścia przekraczającego API.Więcej informacji na temat tego wyjścia zawiera sekcja Wyjście przecięcia funkcji API.

Zestaw znaków i kodowanie

Dane znakowe w produkcie MQXP znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek produktu **CodedCharSetId** . Dane liczbowe w MQXP są kodowane na rodzimym komputerze. Wartość ta jest podawana przez parametr MQENC_NATIVE.

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 537. Pola w MQXP

Nazwa i opis pola	**Nazwa stałej**
StrucId (identyfikator struktury)	MQXP_STRUC_ID (identyfikator struktury MQXC)
Wersja (numer wersji struktury)	MQXP_VERSION_1
ExitId (identyfikator wyjścia)	MQXT_API_CROSSING_EXIT (program zewnętrzny MQXT_API)
ExitReason (przyczyna wywołania wyjścia)	Brak
ExitResponse (odpowiedź z wyjścia)	Brak
ExitCommand (kod wywołania API)	Brak
ExitParmLiczba (liczba parametrów)	Brak
Zarezerwowane (zastrzeżone)	Brak
ExitUserObszar (obszar użytkownika)	Brak

Deklaracje językowe

Deklaracja C dla MQXP

```
typedef struct tagMQXP MQXP;
struct tagMQXP {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQLONG    ExitId;         /* Exit identifier */
  MQLONG    ExitReason;     /* Reason for invocation of exit */
  MQLONG    ExitResponse;   /* Response from exit */
  MQLONG    ExitCommand;    /* API call code */
  MQLONG    ExitParmCount;  /* Parameter count */
  MQLONG    Reserved;       /* Reserved */
  MQBYTE16  ExitUserArea;   /* User area */
};
```

Deklaracja języka COBOL dla MQXP

```
**   MQXP structure
  10 MQXP.
**    Structure identifier
   15 MQXP-STRUCID       PIC X(4).
**    Structure version number
   15 MQXP-VERSION       PIC S9(9) BINARY.
**    Exit identifier
   15 MQXP-EXITID        PIC S9(9) BINARY.
**    Reason for invocation of exit
   15 MQXP-EXITREASON    PIC S9(9) BINARY.
**    Response from exit
   15 MQXP-EXITRESPONSE  PIC S9(9) BINARY.
**    API call code
   15 MQXP-EXITCOMMAND   PIC S9(9) BINARY.
**    Parameter count
   15 MQXP-EXITPARMCOUNT PIC S9(9) BINARY.
**    Reserved
   15 MQXP-RESERVED      PIC S9(9) BINARY.
**    User area
   15 MQXP-EXITUSERAREA  PIC X(16).
```

Deklaracja języka PL/I dla MQXP

```
dcl
 1 MQXP based,
  3 StrucId       char(4),       /* Structure identifier */
  3 Version       fixed bin(31), /* Structure version number */
  3 ExitId        fixed bin(31), /* Exit identifier */
  3 ExitReason    fixed bin(31), /* Reason for invocation of exit */
  3 ExitResponse  fixed bin(31), /* Response from exit */
  3 ExitCommand   fixed bin(31), /* API call code */
  3 ExitParmCount fixed bin(31), /* Parameter count */
  3 Reserved      fixed bin(31), /* Reserved */
  3 ExitUserArea  char(16);      /* User area */
```

Deklaracja High Level Assembler dla MQXP

```
MQXP                DSECT
MQXP_STRUCID        DS   CL4   Structure identifier
MQXP_VERSION        DS   F     Structure version number
MQXP_EXITID         DS   F     Exit identifier
MQXP_EXITREASON     DS   F     Reason for invocation of exit
MQXP_EXITRESPONSE   DS   F     Response from exit
MQXP_EXITCOMMAND    DS   F     API call code
MQXP_EXITPARMCOUNT  DS   F     Parameter count
MQXP_RESERVED       DS   F     Reserved
MQXP_EXITUSERAREA   DS   XL16  User area
*
MQXP_LENGTH         EQU  *-MQXP
                    ORG  MQXP
MQXP_AREA           DS   CL(MQXP_LENGTH)
```

StrucId (MQCHAR4) dla MQXP

Jest to identyfikator struktury struktury parametru wyjścia. Jest to zawsze pole wejściowe. Jego wartością jest MQXP_STRUC_ID.

Wartość musi być następująca:

MQXP_STRUC_ID (identyfikator struktury MQXC)
Identyfikator struktury parametru wyjścia.

Dla języka programowania C zdefiniowana jest również stała MQXP_STRUC_ID_ARRAY. Ma taką samą wartość jak MQXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQXP

Jest to numer wersji struktury. Wartość musi być następująca:

MQXP_VERSION_1
Numer wersji dla struktury bloku parametru wyjścia.

Uwaga: Po wprowadzeniu nowej wersji tej struktury układ istniejącej części nie ulega zmianie. Dlatego wyjście musi sprawdzić, czy numer wersji jest równy lub większy od najniższej wersji zawierającej pola, które mają być używane przez wyjście.

Jest to pole wejściowe do wyjścia.

ExitId (MQLONG) dla MQXP

Ten parametr jest ustawiany przy wejściu do procedury wyjścia i wskazuje typ wyjścia:

MQXT_API_CROSSING_EXIT (program zewnętrzny MQXT_API)
Wyjście przecięcia interfejsu API dla systemu CICS.

Jest to pole wejściowe do wyjścia.

ExitReason (MQLONG) dla MQXP

Jest on ustawiany na wejściu do procedury zewnętrznej. Dla wyjścia przekraczającego API wskazuje, czy procedura jest wywoływana przed, czy po wykonaniu wywołania API:

MQXR_BEFORE,
Przed wykonaniem interfejsu API.

MQXR_PO
Po wykonaniu interfejsu API.

Jest to pole wejściowe do wyjścia.

ExitResponse (MQLONG) dla MQXP

Wartość jest ustawiana przez wyjście w celu komunikacji z programem wywołującym. Zdefiniowane są następujące wartości:

MQXCC_OK
Wyjście zakończone pomyślnie.

MQXCC_SUPPRESS_FUNCTION,
Ukryj funkcję.

Jeśli ta wartość jest ustawiana przez wyjście przecięcia interfejsu API o nazwie *przed* wywołaniem interfejsu API, wywołanie interfejsu API nie jest wykonywane. Zmienna *CompCode* dla wywołania ma wartość MQCC_FAILED, zmienna *Reason* ma wartość MQRC_SUPPRESSED_BY_EXIT, a wszystkie pozostałe parametry pozostają bez zmian.

Jeśli ta wartość jest ustawiana przez wyjście przecięcia interfejsu API o nazwie *po* wywołaniu funkcji API, jest ona ignorowana przez menedżer kolejek.

MQXCC_SKIP_FUNCTION,
Pomiń funkcję.

Jeśli ta wartość jest ustawiana przez wyjście przecięcia interfejsu API o nazwie *przed* wywołaniem funkcji API, wywołanie funkcji API nie jest wykonywane. Parametry *CompCode* i *Reason* oraz wszystkie inne parametry pozostają bez zmian.

Jeśli ta wartość jest ustawiana przez wyjście przecięcia interfejsu API o nazwie *po* wywołaniu funkcji API, jest ona ignorowana przez menedżer kolejek.

Jest to pole wyjściowe wyjścia.

ExitCommand (MQLONG) dla MQXP

To pole jest ustawiane na wejściu do procedury wyjścia. Identyfikuje wywołanie funkcji API, które spowodowało wywołanie wyjścia:

MQXC_CALLBACK,
Wywołanie CALLBACK.

MQXC_MQBACK
Wywołanie MQBACK.

MQXC_MQCB
Wywołanie MQCB.

MQXC_MQCLOSE (MQXC)
Wywołanie MQCLOSE.

MQXC_MQCMIT
Wywołanie MQCMIT.

MQXC_MQCTL,
Wywołanie MQCTL.

MQXC_MQGET (MQXC)
Wywołanie MQGET.

MQXC_MQINQ (kolejka MQXC)
Wywołanie MQINQ.

MQXC_MQOPEN,
Wywołanie MQOPEN.

MQXC_MQPUT
Wywołanie MQPUT.

MQXC_MQPUT1
Wywołanie MQPUT1 .

MQXC_MQSET (zestaw MQXC)
Wywołanie MQSET.

MQXC_MQSTAT,
Wywołanie MQSTAT.

MQXC_MQSUB
Wywołanie MQSUB.

MQXC_MQSUBRQ (kolejka MQXC)
Wywołanie MQSUBRQ.

Jest to pole wejściowe do wyjścia.

ExitParmLicznik (MQLONG) dla MQXP

To pole jest ustawiane na wejściu do procedury wyjścia. Zawiera on liczbę parametrów, które są używane przez wywołanie produktu MQ .

Tabela 538. Liczba parametrów dla każdego wywołania produktu MQ

Nazwa połączenia	Liczba parametrów
MQBACK	3

Tabela 538. Liczba parametrów dla każdego wywołania produktu MQ (kontynuacja)

Nazwa połączenia	Liczba parametrów
MQCLOSE	5
MQCMIT	3
MQGET	9
MQINQ	10
MQOPEN	6
MQPUT	8
MQPUT1	8
MQSET	10

Jest to pole wejściowe do wyjścia.

Zarezerwowany (MQLONG) dla MQXP

Jest to pole zastrzeżone. Jego wartość nie jest istotna dla wyjścia.

Obszar ExitUser(MQBYTE16) dla MQXP

Jest to pole dostępne do użycia przez wyjście. Jest ona inicjowana jako wartość binarna zero dla długości pola przed pierwszym wywołaniem wyjścia dla zadania, a następnie wszystkie zmiany wprowadzone w tym polu przez wyjście są zachowywane między wywołaniami wyjścia. Zdefiniowana jest następująca wartość:

BRAK MQXUA
Brak informacji o użytkowniku.

Wartością długości pola jest zero binarne.

W języku programowania C zdefiniowana jest również stała MQXUA_NONE_ARRAY. Ma ona taką samą wartość jak parametr MQXUA_NONE, ale jest tablicą znaków, a nie łańcuchem.

Długość tego pola jest określona przez wartość MQ_EXIT_USER_AREA_LENGTH. Jest to pole wejściowe/wyjściowe do wyjścia.

MQXQH-nagłówek kolejki transmisji

Struktura MQXQH opisuje informacje, które są poprzedzone danymi komunikatów aplikacji, gdy znajdują się one w kolejkach transmisji.Kolejka transmisji jest specjalnym typem kolejki lokalnej, która tymczasowo przechowuje komunikaty przeznaczone dla kolejek zdalnych (czyli przeznaczone dla kolejek, które nie należą do lokalnego menedżera kolejek). Kolejka transmisji jest oznaczana przez atrybut kolejki **Usage** o wartości MQUS_TRANSMISSION.

Nazwa formatu

MQFMT_XMIT_Q_HEADER

Zestaw znaków i kodowanie

Dane w MQXQH muszą być w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez parametr MQENC_NATIVE.

Ustaw zestaw znaków i kodowanie MQXQH w polach *CodedCharSetId* i *Encoding* w następujących polach:

- Oddzielna struktura MQMD (jeśli struktura MQXQH znajduje się na początku danych komunikatu) lub
- Struktura nagłówka poprzedzająca strukturę MQXQH (wszystkie inne przypadki).

Pola

Uwaga: W poniższej tabeli pola są pogrupowane według użycia, a nie alfabetycznie. Tematy potomne są w tej samej kolejności.

Tabela 539. Pola w MQXQH dla MQXQH

Nazwa i opis pola	Nazwa stałej	Wartość początkowa (jeśli istnieje) stałej
StrucId (identyfikator struktury)	MQXQH_ID_STRUKTURY	'XQH¬'
Wersja (numer wersji struktury)	MQXQH_VERSION_1	1
RemoteQName (nazwa kolejki docelowej)	Brak	Pusty łańcuch lub odstępy
RemoteQMgrNazwa (nazwa docelowego menedżera kolejek)	Brak	Pusty łańcuch lub odstępy
MsgDesc (oryginalny deskryptor komunikatu)	Takie same nazwy i wartości jak w przypadku deskryptora MQMD (patrz sekcja Tabela 500 na stronie 433).	-

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.
3. W języku programowania C: zmienna makraMQXQH_DEFAULT zawiera wartości wymienione w tabeli. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQXQH MyXQH = {MQXQH_DEFAULT};
```

Deklaracje językowe

Deklaracja języka C dla MQXQH

```
typedef struct tagMQXQH MQXQH;
struct tagMQXQH {
  MQCHAR4   StrucId;         /* Structure identifier */
  MQLONG    Version;         /* Structure version number */
  MQCHAR48  RemoteQName;     /* Name of destination queue */
  MQCHAR48  RemoteQMgrName;  /* Name of destination queue manager */
  MQMD1     MsgDesc;         /* Original message descriptor */
};
```

Deklaracja języka COBOL dla MQXQH

```
**   MQXQH structure
  10 MQXQH.
**    Structure identifier
   15 MQXQH-STRUCID                  PIC X(4).
**    Structure version number
   15 MQXQH-VERSION                  PIC S9(9) BINARY.
**    Name of destination queue
   15 MQXQH-REMOTEQNAME              PIC X(48).
**    Name of destination queue manager
   15 MQXQH-REMOTEQMGRNAME           PIC X(48).
**    Original message descriptor
   15 MQXQH-MSGDESC.
```

```
**     Structure identifier
    20 MQXQH-MSGDESC-STRUCID          PIC X(4).
**     Structure version number
    20 MQXQH-MSGDESC-VERSION          PIC S9(9) BINARY.
**     Report options
    20 MQXQH-MSGDESC-REPORT           PIC S9(9) BINARY.
**     Message type
    20 MQXQH-MSGDESC-MSGTYPE          PIC S9(9) BINARY.
**     Expiry time
    20 MQXQH-MSGDESC-EXPIRY           PIC S9(9) BINARY.
**     Feedback or reason code
    20 MQXQH-MSGDESC-FEEDBACK         PIC S9(9) BINARY.
**     Numeric encoding of message data
    20 MQXQH-MSGDESC-ENCODING         PIC S9(9) BINARY.
**     Character set identifier of message data
    20 MQXQH-MSGDESC-CODEDCHARSETID   PIC S9(9) BINARY.
**     Format name of message data
    20 MQXQH-MSGDESC-FORMAT           PIC X(8).
**     Message priority
    20 MQXQH-MSGDESC-PRIORITY         PIC S9(9) BINARY.
**     Message persistence
    20 MQXQH-MSGDESC-PERSISTENCE      PIC S9(9) BINARY.
**     Message identifier
    20 MQXQH-MSGDESC-MSGID            PIC X(24).
**     Correlation identifier
    20 MQXQH-MSGDESC-CORRELID         PIC X(24).
**     Backout counter
    20 MQXQH-MSGDESC-BACKOUTCOUNT     PIC S9(9) BINARY.
**     Name of reply-to queue
    20 MQXQH-MSGDESC-REPLYTOQ         PIC X(48).
**     Name of reply queue manager
    20 MQXQH-MSGDESC-REPLYTOQMGR      PIC X(48).
**     User identifier
    20 MQXQH-MSGDESC-USERIDENTIFIER   PIC X(12).
**     Accounting token
    20 MQXQH-MSGDESC-ACCOUNTINGTOKEN  PIC X(32).
**     Application data relating to identity
    20 MQXQH-MSGDESC-APPLIDENTITYDATA PIC X(32).
**     Type of application that put the message
    20 MQXQH-MSGDESC-PUTAPPLTYPE      PIC S9(9) BINARY.
**     Name of application that put the message
    20 MQXQH-MSGDESC-PUTAPPLNAME      PIC X(28).
**     Date when message was put
    20 MQXQH-MSGDESC-PUTDATE          PIC X(8).
**     Time when message was put
    20 MQXQH-MSGDESC-PUTTIME          PIC X(8).
**     Application data relating to origin
    20 MQXQH-MSGDESC-APPLORIGINDATA   PIC X(4).
```

Deklaracja języka PL/I dla MQXQH

```
dcl
 1 MQXQH based,
  3 StrucId           char(4),       /* Structure identifier */
  3 Version           fixed bin(31), /* Structure version number */
  3 RemoteQName       char(48),      /* Name of destination queue */
  3 RemoteQMgrName    char(48),      /* Name of destination queue
                                        manager */
  3 MsgDesc,                         /* Original message descriptor */
   5 StrucId          char(4),       /* Structure identifier */
   5 Version          fixed bin(31), /* Structure version number */
   5 Report           fixed bin(31), /* Report options */
   5 MsgType          fixed bin(31), /* Message type */
   5 Expiry           fixed bin(31), /* Expiry time */
   5 Feedback         fixed bin(31), /* Feedback or reason code */
   5 Encoding         fixed bin(31), /* Numeric encoding of message
                                        data */
   5 CodedCharSetId   fixed bin(31), /* Character set identifier of
                                        message data */
   5 Format           char(8),       /* Format name of message data */
   5 Priority         fixed bin(31), /* Message priority */
   5 Persistence      fixed bin(31), /* Message persistence */
   5 MsgId            char(24),      /* Message identifier */
   5 CorrelId         char(24),      /* Correlation identifier */
   5 BackoutCount     fixed bin(31), /* Backout counter */
   5 ReplyToQ         char(48),      /* Name of reply-to queue */
   5 ReplyToQMgr      char(48),      /* Name of reply queue manager */
   5 UserIdentifier   char(12),      /* User identifier */
   5 AccountingToken  char(32),      /* Accounting token */
```

```
   5 ApplIdentityData char(32),      /* Application data relating to
                                        identity */
   5 PutApplType      fixed bin(31), /* Type of application that put the
                                        message */
   5 PutApplName      char(28),      /* Name of application that put the
                                        message */
   5 PutDate          char(8),       /* Date when message was put */
   5 PutTime          char(8),       /* Time when message was put */
   5 ApplOriginData   char(4);       /* Application data relating to
                                        origin */
```

Deklaracja High Level Assembler dla MQXQH

```
MQXQH                           DSECT
MQXQH_STRUCID                   DS   CL4   Structure identifier
MQXQH_VERSION                   DS   F     Structure version number
MQXQH_REMOTEQNAME               DS   CL48  Name of destination queue
MQXQH_REMOTEQMGRNAME            DS   CL48  Name of destination queue
*                                          manager
MQXQH_MSGDESC                   DS   0F    Force fullword alignment
MQXQH_MSGDESC_STRUCID           DS   CL4   Structure identifier
MQXQH_MSGDESC_VERSION           DS   F     Structure version number
MQXQH_MSGDESC_REPORT            DS   F     Report options
MQXQH_MSGDESC_MSGTYPE           DS   F     Message type
MQXQH_MSGDESC_EXPIRY            DS   F     Expiry time
MQXQH_MSGDESC_FEEDBACK          DS   F     Feedback or reason code
MQXQH_MSGDESC_ENCODING          DS   F     Numeric encoding of message
*                                          data
MQXQH_MSGDESC_CODEDCHARSETID    DS   F     Character set identifier of
*                                          message data
MQXQH_MSGDESC_FORMAT            DS   CL8   Format name of message data
MQXQH_MSGDESC_PRIORITY          DS   F     Message priority
MQXQH_MSGDESC_PERSISTENCE       DS   F     Message persistence
MQXQH_MSGDESC_MSGID             DS   XL24  Message identifier
MQXQH_MSGDESC_CORRELID          DS   XL24  Correlation identifier
MQXQH_MSGDESC_BACKOUTCOUNT      DS   F     Backout counter
MQXQH_MSGDESC_REPLYTOQ          DS   CL48  Name of reply-to queue
MQXQH_MSGDESC_REPLYTOQMGR       DS   CL48  Name of reply queue manager
MQXQH_MSGDESC_USERIDENTIFIER    DS   CL12  User identifier
MQXQH_MSGDESC_ACCOUNTINGTOKEN   DS   XL32  Accounting token
MQXQH_MSGDESC_APPLIDENTITYDATA  DS   CL32  Application data relating to
*                                          identity
MQXQH_MSGDESC_PUTAPPLTYPE       DS   F     Type of application that put
*                                          the message
MQXQH_MSGDESC_PUTAPPLNAME       DS   CL28  Name of application that put
*                                          the message
MQXQH_MSGDESC_PUTDATE           DS   CL8   Date when message was put
MQXQH_MSGDESC_PUTTIME           DS   CL8   Time when message was put
MQXQH_MSGDESC_APPLORIGINDATA    DS   CL4   Application data relating to
*                                          origin
MQXQH_MSGDESC_LENGTH            EQU  *-MQXQH_MSGDESC
                                ORG  MQXQH_MSGDESC
MQXQH_MSGDESC_AREA              DS   CL(MQXQH_MSGDESC_LENGTH)
*
MQXQH_LENGTH                    EQU  *-MQXQH
                                ORG  MQXQH
MQXQH_AREA                      DS   CL(MQXQH_LENGTH)
```

Deklaracja Visual Basic dla MQXQH

```
Type MQXQH
  StrucId        As String*4  'Structure identifier'
  Version        As Long      'Structure version number'
  RemoteQName    As String*48 'Name of destination queue'
  RemoteQMgrName As String*48 'Name of destination queue manager'
  MsgDesc        As MQMD1     'Original message descriptor'
End Type
```

Pola w oddzielnym deskryptorze komunikatu

Komunikat znajdujący się w kolejce transmisji ma *dwa* deskryptory komunikatów:

- Jeden deskryptor komunikatu jest przechowywany oddzielnie od danych komunikatu. Jest on nazywany *oddzielnym deskryptorem komunikatu*i jest generowany przez menedżer kolejek, gdy komunikat jest

umieszczany w kolejce transmisji. Niektóre pola w oddzielnym deskryptorze komunikatu są kopiowane z deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Osobny deskryptor komunikatu jest zwracany do aplikacji w parametrze **MsgDesc** wywołania MQGET, gdy komunikat jest usuwany z kolejki transmisji.

- Drugi deskryptor komunikatu jest przechowywany w strukturze MQXQH jako część danych komunikatu. Jest on nazywany *osadzonym deskryptorem komunikatu*i jest kopią deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 (z niewielkimi wariantami).

  Osadzonym deskryptorem komunikatu jest zawsze deskryptor MQMD version-1 . Jeśli komunikat umieszczony przez aplikację ma wartości inne niż domyślne dla jednego lub większej liczby pól version-2 w deskryptorze MQMD, struktura MQMDE jest zgodna z MQXQH i po niej następują dane komunikatu aplikacji (jeśli istnieją). Produkt MQMDE może mieć jedną z następujących wartości:

  – Wygenerowane przez menedżer kolejek (jeśli do umieszczenia komunikatu aplikacja używa deskryptora MQMD w wersji version-2) lub
  – Istnieje już na początku danych komunikatu aplikacji (jeśli do umieszczenia komunikatu aplikacja używa programu MQMD w wersji version-1).

  Osadzony deskryptor komunikatu jest to deskryptor, który jest zwracany do aplikacji w parametrze **MsgDesc** wywołania MQGET, gdy komunikat jest usuwany z końcowej kolejki docelowej.

Pola w oddzielnym deskryptorze komunikatu są ustawiane przez menedżer kolejek w przedstawiony sposób. Jeśli menedżer kolejek nie obsługuje deskryptora MQMD version-2 , używany jest deskryptor MQMD version-1 bez utraty funkcji.

Tabela 540. Wartości używane dla pól w oddzielnej strukturze MQMD

Pole w oddzielnej strukturze MQMD	Użyta wartość
StrucId	MQMD_STRUC_ID (Identyfikator struktury kolejki)
Version	MQMD_VERSION_2
Report	Kopiowane z osadzonego deskryptora komunikatu, ale z bitami identyfikowanymi przez parametr MQRO_ACCEPT_UNSUP_IF_XMIT_MASK ustawionymi na zero. Zapobiega to generowaniu komunikatu raportu COA lub COD, gdy komunikat jest umieszczany w kolejce transmisji lub z niej usuwany.
MsgType	Skopiowane z osadzonego deskryptora komunikatu.
Expiry	Skopiowane z osadzonego deskryptora komunikatu.
Feedback	Skopiowane z osadzonego deskryptora komunikatu.
Encoding	MQENC_NATIVE (patrz uwaga)
CodedCharSetId	Atrybut **CodedCharSetId** menedżera kolejek.
Format	MQFMT_XMIT_Q_HEADER
Priority	Skopiowane z osadzonego deskryptora komunikatu.
Persistence	Skopiowane z osadzonego deskryptora komunikatu.
MsgId	Nowa wartość jest generowana przez menedżer kolejek. Ten identyfikator komunikatu różni się od identyfikatora komunikatu *MsgId* , który mógł zostać wygenerowany przez menedżer kolejek dla opisanego wcześniej osadzonego deskryptora komunikatu.
CorrelId	Wartość *MsgId* z osadzonego deskryptora komunikatu. Dla komunikatów umieszczanych w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE, *CorrelId* jest zarezerwowany do użytku wewnętrznego.
BackoutCount	0

Tabela 540. Wartości używane dla pól w oddzielnej strukturze MQMD (kontynuacja)

Pole w oddzielnej strukturze MQMD	Użyta wartość
ReplyToQ	Skopiowane z osadzonego deskryptora komunikatu.
ReplyToQMgr	Skopiowane z osadzonego deskryptora komunikatu.
UserIdentifier	Skopiowane z osadzonego deskryptora komunikatu.
AccountingToken	Skopiowane z osadzonego deskryptora komunikatu. Dla komunikatów umieszczanych w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE, *AccountingToken* jest zarezerwowany do użytku wewnętrznego.
ApplIdentityData	Skopiowane z osadzonego deskryptora komunikatu.
PutApplType	MQAT_QMGR
PutApplName	Pierwsze 28 bajtów nazwy menedżera kolejek.
PutDate	Data umieszczenia komunikatu w kolejce transmisji.
PutTime	Czas umieszczenia komunikatu w kolejce transmisji.
ApplOriginData	Puste
GroupId	MQGI_NONE
MsgSeqNumber	1
Offset	0
MsgFlags	MQMF_BRAK
OriginalLength	MQOL_UNDEFINED

- W systemie Windowswartość zmiennej MQENC_NATIVE dla Micro Focus COBOL różni się od wartości zmiennej C. Wartość w polu *Encoding* w oddzielnym deskryptorze komunikatu jest zawsze wartością dla języka C w tych środowiskach. Wartość ta jest liczbą dziesiętną wynoszącą 546. Ponadto pola liczb całkowitych w strukturze MQXQH mają kodowanie odpowiadające tej wartości (rodzime kodowanie Intel).

Pola w osadzonym deskryptorze komunikatu

Pola w osadzonym deskryptorze komunikatu mają takie same wartości jak pola w parametrze **MsgDesc** wywołania MQPUT lub MQPUT1 , z wyjątkiem następujących:

- Pole *Version* zawsze ma wartość MQMD_VERSION_1.
- Jeśli pole *Priority* ma wartość MQPRI_PRIORITY_AS_Q_DEF, jest ono zastępowane wartością atrybutu **DefPriority** kolejki.
- Jeśli pole *Persistence* ma wartość MQPER_PERSISTENCE_AS_Q_DEF, jest ono zastępowane wartością atrybutu **DefPersistence** kolejki.
- Jeśli pole *MsgId* ma wartość MQMI_NONE, podano opcję MQPMO_NEW_MSG_ID lub komunikat jest komunikatem listy dystrybucyjnej, wartość *MsgId* jest zastępowana nowym identyfikatorem komunikatu wygenerowanym przez menedżer kolejek.

  Gdy komunikat listy dystrybucyjnej jest dzielony na mniejsze komunikaty listy dystrybucyjnej umieszczone w różnych kolejkach transmisji, pole *MsgId* w każdym z nowych osadzonych deskryptorów komunikatów jest takie samo jak w oryginalnym komunikacie listy dystrybucyjnej.
- Jeśli określono opcję MQPMO_NEW_CORREL_ID, identyfikator *CorrelId* jest zastępowany nowym identyfikatorem korelacji wygenerowanym przez menedżer kolejek.
- Pola kontekstu są ustawiane zgodnie z opcjami MQPMO_ * _CONTEXT określonymi w parametrze **PutMsgOpts** . Pola kontekstu są następujące:

  - *AccountingToken*
  - *ApplIdentityData*
  - *ApplOriginData*
  - *PutApplName*
  - *PutApplType*
  - *PutDate*
  - *PutTime*
  - *UserIdentifier*
- Pola version-2 (jeśli były obecne) są usuwane z deskryptora MQMD i przenoszone do struktury MQMDE, jeśli co najmniej jedno z pól version-2 ma wartość inną niż domyślna.

Umieszczanie komunikatów w kolejkach zdalnych

Gdy aplikacja umieszcza komunikat w kolejce zdalnej (poprzez bezpośrednie określenie nazwy kolejki zdalnej lub użycie lokalnej definicji kolejki zdalnej), menedżer kolejek lokalnych:

- Tworzy strukturę MQXQH zawierającą osadzony deskryptor komunikatu
- Dołącza MQMDE, jeśli jest potrzebny i nie jest jeszcze obecny
- Dołącza dane komunikatu aplikacji
- Umieszcza komunikat w odpowiedniej kolejce transmisji

Umieszczanie komunikatów bezpośrednio w kolejkach transmisji

Aplikacja może również umieścić komunikat bezpośrednio w kolejce transmisji. W takim przypadku aplikacja musi poprzedzać dane komunikatu aplikacji strukturą MQXQH i inicjować pola odpowiednimi wartościami. Ponadto pole *Format* w parametrze **MsgDesc** wywołania MQPUT lub MQPUT1 musi mieć wartość MQFMT_XMIT_Q_HEADER.

Dane znakowe w strukturze MQXQH utworzonej przez aplikację muszą znajdować się w zestawie znaków lokalnego menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek **CodedCharSetId**), a dane całkowite muszą być zakodowane na komputerze rodzimym. Ponadto dane znakowe w strukturze MQXQH muszą być dopełnione odstępami do zdefiniowanej długości pola. Dane nie mogą zostać przedwcześnie zakończone przy użyciu znaku o kodzie zero, ponieważ menedżer kolejek nie przekształca znaków o kodzie zero i kolejnych znaków w znaki o kodzie zero w strukturze MQXQH.

Jednak menedżer kolejek nie sprawdza, czy istnieje struktura MQXQH lub czy określono poprawne wartości dla pól.

Aplikacje nie powinny umieszczać swoich komunikatów bezpośrednio w systemie SYSTEM.CLUSTER.TRANSMIT.QUEUE.

Pobieranie komunikatów z kolejek transmisji

Aplikacje, które pobierają komunikaty z kolejki transmisji, muszą przetwarzać informacje w strukturze MQXQH w odpowiedni sposób. Obecność struktury MQXQH na początku danych komunikatu aplikacji jest wskazywana przez wartość MQFMT_XMIT_Q_HEADER zwracaną w polu *Format* parametru **MsgDesc** wywołania MQGET. Wartości zwracane w polach *CodedCharSetId* i *Encoding* parametru **MsgDesc** wskazują zestaw znaków i kodowanie danych znakowych i całkowitych w strukturze MQXQH. Zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane przez pola *CodedCharSetId* i *Encoding* w osadzonym deskryptorze komunikatu.

StrucId (MQCHAR4) dla MQXQH

Jest to identyfikator struktury struktury nagłówka kolejki transmisji. Jest to zawsze pole wejściowe. Jego wartością jest MQXQH_STRUC_ID.

Wartość musi być następująca:

MQXQH_ID_STRUKTURY
Identyfikator struktury nagłówka kolejki transmisji.

Dla języka programowania C zdefiniowana jest również stała MQXQH_STRUC_ID_ARRAY. Ma taką samą wartość jak MQXQH_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja (MQLONG) dla MQXQH

Jest to numer wersji struktury. Wartość musi być następująca:

MQXQH_VERSION_1
Numer wersji struktury nagłówka kolejki transmisji.

Następująca stała określa numer wersji bieżącej:

MQXQH_CURRENT_VERSION
Bieżąca wersja struktury nagłówka kolejki transmisji.

Wartością początkową tego pola jest MQXQH_VERSION_1.

RemoteQName (MQCHAR48) dla MQXQH

Jest to nazwa kolejki komunikatów, która jest widocznym miejscem docelowym komunikatu (może się okazać, że nie jest to miejsce docelowe, jeśli na przykład ta kolejka jest zdefiniowana w pliku *RemoteQMgrName* jako lokalna definicja innej kolejki zdalnej).

Jeśli komunikat jest komunikatem listy dystrybucyjnej (pole *Format* w deskryptorze osadzonego komunikatu ma wartość MQFMT_DIST_HEADER), pole *RemoteQName* jest puste.

Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

RemoteQMgrNazwa (MQCHAR48) dla MQXQH

Jest to nazwa menedżera kolejek lub grupy współużytkowania kolejek, która jest właścicielem kolejki będącej pozornym miejscem docelowym komunikatu.

Jeśli komunikat jest komunikatem listy dystrybucyjnej, pole *RemoteQMgrName* jest puste.

Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania.

MsgDesc (MQMD1) dla MQXQH

Jest to osadzony deskryptor komunikatu i jest to bliska kopia deskryptora komunikatu MQMD, który został określony jako parametr **MsgDesc** w wywołaniu MQPUT lub MQPUT1 podczas pierwotnego umieszczania komunikatu w kolejce zdalnej.

Uwaga: Jest to komenda MQMD w wersji version-1 .

Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQMD.

Wywołania funkcji

Ta sekcja zawiera informacje o wszystkich możliwych wywołaniach MQI. Opisy, składnia, informacje o parametrach, uwagi dotyczące składni i wywołania języka dla każdego z możliwych języków są podane dla każdego z różnych wywołań.

Odsyłacze pokrewne

z/OS Przykłady danych wyjściowych CEDF z wywołań MQI

Opisy połączeń

W tej sekcji opisano wywołania MQI.

- “MQBACK-wycofanie zmian” na stronie 647

- “MQBEGIN-Rozpoczęcie jednostki pracy” na stronie 651
- “MQBUFMH-Przekształć bufor w uchwyt komunikatu” na stronie 654
- “MQCB-zarządzanie wywołaniem zwrotnym” na stronie 658
- “MQCB_FUNCTION-funkcja wywołania zwrotnego” na stronie 668
- “MQCLOSE-zamknięcie obiektu” na stronie 669
- “MQCMIT-zatwierdzanie zmian” na stronie 678
- “MQCONN-Połącz z menedżerem kolejek” na stronie 682
- “MQCONNX-Połącz z menedżerem kolejek (rozszerzone)” na stronie 689
- “MQCRTMH-Tworzenie uchwytu komunikatu” na stronie 695
- “MQCTL-wywołania zwrotne sterowania” na stronie 699
- “MQDISC-Rozłączanie menedżera kolejek” na stronie 705
- “MQDLTMH-Usuwanie uchwytu komunikatu” na stronie 709
- “MQDLTMP-właściwość usuwania komunikatu” na stronie 712
- “MQGET-pobierz komunikat” na stronie 714
- “MQINQ-zapytanie o obiekt-atrybuty” na stronie 727
- “MQINQMP-właściwość komunikatu zapytania” na stronie 746
- “MQMHBUF-przekształcenie uchwytu komunikatu w bufor” na stronie 752
- “MQOPEN-otwarcie obiektu” na stronie 756
- “MQPUT-komunikat umieszczania” na stronie 774
- “MQPUT1 -Umieść jeden komunikat” na stronie 788
- “MQSET-ustawianie atrybutów obiektu” na stronie 799
- “MQSETMP-ustawianie właściwości komunikatu” na stronie 805
- “MQSTAT-pobieranie informacji o statusie” na stronie 809
- “MQMHBUF-przekształcenie uchwytu komunikatu w bufor” na stronie 752
- “MQSUB-rejestrowanie subskrypcji” na stronie 813
- “MQSUBRQ-żądanie subskrypcji” na stronie 821

Dla tych wywołań dostępna jest pomoc elektroniczna na platformach UNIX w postaci stron podręcznika (*man*).

Uwaga: Wywołania powiązane z konwersją danych MQXCNVC i MQ_DATA_CONV_EXIT znajdują się w pliku “Wyjście konwersji danych” na stronie 937.

Konwencje używane w opisach wywołań

Dla każdego wywołania ta kolekcja tematów zawiera opis parametrów i użycia wywołania w formacie, który jest niezależny od języka programowania. Po tym następują typowe wywołania wywołania wywołania i typowe deklaracje jego parametrów w każdym z obsługiwanych języków programowania.

Ważne: Podczas kodowania wywołań funkcji API języka IBM MQ należy upewnić się, że zostały podane wszystkie istotne parametry (zgodnie z opisem w poniższych sekcjach). Brak takiego działania może spowodować nieprzewidywalne rezultaty.

Opis każdego wywołania zawiera następujące sekcje:

Nazwa połączenia
Nazwa połączenia, po której następuje krótki opis przeznaczenia połączenia.

Parametry
Dla każdego parametru po nazwie występuje jego typ danych w nawiasach () i jedną z następujących:

dane wejściowe
Informacje są podawane w parametrze podczas wykonywania wywołania.

wyniki
Menedżer kolejek zwraca informacje w parametrze po zakończeniu lub niepowodzeniu wywołania.

Wejście/wyjście
Informacje są podawane w parametrze podczas wykonywania wywołania, a menedżer kolejek zmienia informacje po zakończeniu lub niepowodzeniu wywołania.

Na przykład:

Compcode (MQLONG)-dane wyjściowe

W niektórych przypadkach typem danych jest struktura. We wszystkich przypadkach istnieje więcej informacji na temat typu danych lub struktury w programie "Podstawowe typy danych" na stronie 236.

Dwa ostatnie parametry w każdym wywołaniu są kodem zakończenia i kodem przyczyny. Kod zakończenia wskazuje, czy wywołanie zakończyło się pomyślnie, częściowo, czy wcale. Więcej informacji o częściowym powodzeniu lub niepowodzeniu wywołania znajduje się w kodzie przyczyny. Więcej informacji na temat każdego kodu zakończenia i przyczyny zawiera sekcja "Kody powrotu" na stronie 903.

Użycie notatek
Dodatkowe informacje o wywołaniu, opisujące sposób jego użycia oraz wszelkie ograniczenia dotyczące jego użycia.

Wywołanie języka asemblera
Typowe wywołanie wywołania i deklaracja jego parametrów w języku asemblera.

Wywołanie C
Typowe wywołanie wywołania i deklaracja jego parametrów w języku C.

Wywołanie COBOL
Typowe wywołanie wywołania i deklaracja jego parametrów w języku COBOL.

Wywołanie PL/I
Typowe wywołanie wywołania i deklaracja jego parametrów w PL/I.

Wszystkie parametry są przekazywane przez referencję.

Wywołanie Visual Basic
Typowe wywołanie wywołania i deklaracja jego parametrów w języku Visual Basic.

Inne konwencje notacji to:

Stałe
Nazwy stałych są wyświetlane wielkimi literami, na przykład MQOO_OUTPUT. Zestaw stałych o tym samym przedrostku jest wyświetlany w następujący sposób: MQIA_ *. Wartość stałej można znaleźć w sekcji "Stałe" na stronie 61 .

Tablice
W niektórych wywołaniach parametry są tablicami łańcuchów znaków, które nie mają stałej wielkości. W opisach tych parametrów mała litera n oznacza stałą numeryczną. Podczas kodowania deklaracji dla tego parametru należy zastąpić wartość n wymaganą wartością liczbową.

Korzystanie z wywołań w języku C

Parametry *tylko wejściowe* i typu MQHCONN, MQHOBJ, MQHMSG lub MQLONG są przekazywane przez wartość. W przypadku wszystkich innych parametrów *adres* parametru jest przekazywany przez wartość.

Nie trzeba podawać wszystkich parametrów przekazywanych przez adres przy każdym wywołaniu funkcji. Jeśli konkretny parametr nie jest potrzebny, należy określić wskaźnik pusty jako parametr w wywołaniu funkcji zamiast adresu danych parametru. Parametry, dla których jest to możliwe, są określone w opisach wywołań.

Jako wartość wywołania nie jest zwracany żaden parametr; w terminologii C oznacza to, że wszystkie wywołania zwracają wartość `void`.

Deklarowanie parametru Buffer

Wywołania **MQGET**, **MQPUT**i **MQPUT1** mają jeden parametr, który ma niezdefiniowany typ danych: parametr *Buffer* . Ten parametr służy do wysyłania i odbierania danych komunikatu aplikacji.

Parametry tego sortowania są wyświetlane w przykładach w języku C jako tablice MQBYTE. W ten sposób można zadeklarować parametry, ale zwykle wygodniej jest zadeklarować je jako konkretną strukturę, która opisuje układ danych w komunikacie. Prototyp funkcji deklaruje parametr jako wskaźnik do unieważnienia, dzięki czemu można określić adres dowolnego rodzaju danych jako parametr w wywołaniu.

Wskaźnik do-void jest wskaźnikiem do danych w niezdefiniowanym formacie. Jest on zdefiniowany jako:

```
typedef void *PMQVOID;
```

MQBACK-wycofanie zmian

Wywołanie MQBACK wskazuje menedżerowi kolejek, że wszystkie operacje pobierania i umieszczania komunikatów, które wystąpiły od ostatniego punktu synchronizacji, mają zostać wycofane.

Komunikaty umieszczone jako część jednostki pracy są usuwane; komunikaty pobrane jako część jednostki pracy są przywracane do kolejki.

- W systemie z/OSto wywołanie jest używane tylko przez programy wsadowe (w tym programy w języku IMS).

Składnia

MQBACK (*Hconn*, *Compcode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

Kod zakończenia
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_OUTCOME_PENDING (oczekiwanie na wynik MQ)
(2124, X'84C') Wynik operacji wycofanych danych jest w toku.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

MQRC_OUTCOME_MIXED
(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn .

Użycie notatek

1. Tego wywołania można użyć tylko wtedy, gdy sam menedżer kolejek koordynuje jednostkę pracy. Może to być:
   - Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby produktu MQ .
   - Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby produktu MQ .

   Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja “MQBEGIN- Rozpoczęcie jednostki pracy” na stronie 651.
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiedniego wywołania wycofania zamiast wywołania MQBACK. Środowisko może również obsługiwać niejawne wycofanie spowodowane nieprawidłowym zakończeniem działania aplikacji.
   - W systemie z/OSnależy użyć następujących wywołań:
     – Programy wsadowe (w tym programy IMS w języku DL/I) mogą używać wywołania MQBACK, jeśli jednostka pracy dotyczy tylko zasobów MQ . Jeśli jednak jednostka pracy ma wpływ zarówno na zasoby produktu MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRBACK udostępnianego przez usługę RRS (z/OS Recoverable Resource

Service). Wywołanie SRRBACK wycofuje zmiany w zasobach należących do menedżerów zasobów, dla których włączono koordynację RRS.

- Aplikacje CICS muszą używać komendy `EXEC CICS SYNCPOINT ROLLBACK` , aby wycofać jednostkę pracy. Nie należy używać wywołania MQBACK dla aplikacji CICS .
- Aplikacje IMS (inne niż programy wsadowe DL/I) muszą używać wywołań języka IMS , takich jak `ROLB` , do tworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji IMS (innych niż programy wsadowe DL/I).

- W systemie IBM itego wywołania należy używać dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie została wydana dla zadania.

3. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji "MQDISC-Rozłączanie menedżera kolejek" na stronie 705 .
4. Gdy aplikacja umieszcza lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Te informacje są powiązane z uchwytem kolejki i obejmują takie elementy, jak:

   - Wartości pól *GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags* w strukturze MQMD.
   - Określa, czy komunikat jest częścią jednostki pracy.
   - Dla wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

   Menedżer kolejek przechowuje *trzy* zestawy informacji o grupach i segmentach, po jednym zestawie dla każdego z następujących zestawów:

   - Ostatnie pomyślne wywołanie MQPUT (może być częścią jednostki pracy).
   - Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
   - Ostatnie pomyślne wywołanie MQGET, które przeglądnięto komunikat w kolejce (nie może być częścią jednostki pracy).

5. Informacje powiązane z wywołaniem MQGET zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

   Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zasięgiem jednostki pracy, nie mają odtworzonych informacji o grupach i segmentach, jeśli jednostka pracy została wycofana.

   Przywrócenie poprzedniej wartości informacji o grupie i segmencie po wycofaniu jednostki pracy umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów na kilka jednostek pracy oraz zrestartowanie w odpowiednim punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się.

   Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną pamięć kolejki. Jednak aplikacja musi zachować wystarczającą ilość informacji, aby w przypadku awarii systemu możliwe było zrestartowanie umieszczania lub pobierania komunikatów w odpowiednim miejscu.

   Szczegółowe informacje na temat restartowania w odpowiednim momencie po awarii systemu zawiera opis opcji MQPMO_LOGICAL_ORDER w sekcji "MQPMO-opcje umieszczania komunikatów" na stronie 514oraz opis opcji MQGMO_LOGICAL_ORDER w sekcji "MQGMO-opcje pobierania komunikatów" na stronie 376.

   Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy.

6. Jednostka pracy ma taki sam zasięg jak uchwyt połączenia. Wszystkie wywołania produktu MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wykonane przy użyciu innego uchwytu połączenia (na przykład wywołania wykonane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączeń zawiera opis parametru **Hconn** w sekcji "MQCONN-Połącz z menedżerem kolejek" na stronie 682 .
7. To wywołanie ma wpływ tylko na komunikaty, które zostały umieszczone lub pobrane jako część bieżącej jednostki pracy.
8. Aplikacja długotrwała, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w jednostce pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może zapełniać kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć się przed taką możliwością, administrator musi ustawić atrybut **MaxUncommittedMsgs** menedżera kolejek na wartość, która jest na tyle niska, aby zapobiec zapełnianiu kolejek przez aplikacje niedziałające, ale na tyle wysoka, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Wywołanie C

```
MQBACK (Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Wywołanie COBOL

```
CALL 'MQBACK' USING HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQBACK (Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Wywołanie programu High Level Assembler

```
CALL MQBACK,(HCONN,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Wywołanie Visual Basic

```
MQBACK Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQBEGIN-Rozpoczęcie jednostki pracy

Wywołanie MQBEGIN rozpoczyna jednostkę pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zewnętrzne menedżery zasobów.

Składnia

MQBEGIN (*Hconn*, *BeginOptions*, *Compcode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

Hconn musi być niewspółużytkowanym uchwytem połączenia. Jeśli określono uchwyt połączenia współużytkowanego, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_HCONN_ERROR. Więcej informacji na temat współużytkowanych i niewspółużytkowanych uchwytów zawiera opis opcji MQCNO_HANDLE_SHARE_ * w sekcji “MQCNO-opcje połączenia” na stronie 321 .

BeginOptions
Typ: MQBO-input/output

Są to opcje, które sterują działaniem komendy MQBEGIN zgodnie z opisem w sekcji “MQBO-opcje początku” na stronie 283.

Jeśli nie są wymagane żadne opcje, programy napisane w języku C lub S/390 mogą określać adres parametru o wartości NULL zamiast określać adres struktury MQBO.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna

Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_NO_EXTERNAL_UCZESTNICY
(2121, X'849 ') Brak zarejestrowanych uczestniczących menedżerów zasobów.

MQRC_XX_ENCODE_CASE_ONE partia_cząstkowa NIEDOSTĘPNA
(2122, X'84A') Uczestniczy menedżer zasobów jest niedostępny.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

MQRC_BO_BŁĄD
(2134, X'856 ') Niepoprawna struktura opcji początku.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_IN_PROGRESS
(2128, X'850 ') Jednostka pracy została już uruchomiona.

Więcej informacji na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Użyj wywołania MQBEGIN, aby uruchomić jednostkę pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zmiany w zasobach należących do innych menedżerów zasobów. Menedżer kolejek obsługuje trzy typy jednostek pracy:
   - **Lokalna jednostka pracy koordynowana przez menedżer kolejek**: Jednostka pracy, w której menedżer kolejek jest jedynym uczestniczącym menedżerem zasobów, dlatego menedżer kolejek pełni rolę koordynatora jednostki pracy.
     - Aby uruchomić ten typ jednostki pracy, należy określić opcję MQPMO_SYNCPOINT lub MQGMO_SYNCPOINT w pierwszym wywołaniu MQPUT, MQPUT1lub MQGET w jednostce pracy.
     - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołania MQCMIT lub MQBACK.

- **Menedżer kolejek-skoordynowana globalna jednostka pracy**: jednostka pracy, w której menedżer kolejek pełni rolę koordynatora jednostki pracy, zarówno dla MQ zasobów *i* , jak i dla zasobów należących do innych menedżerów zasobów. Te menedżery zasobów współpracują z menedżerem kolejek w celu zapewnienia, że wszystkie zmiany zasobów w jednostce pracy zostaną zatwierdzone lub wycofane razem.
  - Aby uruchomić ten typ jednostki pracy, należy użyć wywołania MQBEGIN.
  - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań MQCMIT i MQBACK.
- **Globalna jednostka pracy koordynowana zewnętrznie**: jednostka pracy, w której menedżer kolejek jest uczestnikiem, ale menedżer kolejek nie działa jako koordynator jednostki pracy. Zamiast tego istnieje zewnętrzny koordynator jednostki pracy, z którym współpracuje menedżer kolejek.
  - Aby uruchomić ten typ jednostki pracy, należy użyć odpowiedniego wywołania udostępnionego przez zewnętrznego koordynatora jednostki pracy.

    Jeśli wywołanie MQBEGIN jest używane do próby uruchomienia jednostki pracy, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ENVIRONMENT_ERROR.
  - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań zatwierdzania i wycofanych udostępnianych przez zewnętrznego koordynatora jednostki pracy.

    Jeśli do zatwierdzenia lub wycofania jednostki pracy używane jest wywołanie MQCMIT lub MQBACK, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ENVIRONMENT_ERROR.

2. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji "MQDISC-Rozłączanie menedżera kolejek" na stronie 705 .
3. Aplikacja może jednocześnie uczestniczyć tylko w jednej jednostce pracy. Wywołanie komendy MQBEGIN kończy się niepowodzeniem z kodem przyczyny MQRC_UOW_IN_PROGRESS, jeśli dla aplikacji istnieje już jednostka pracy, niezależnie od typu jednostki pracy.
4. Wywołanie MQBEGIN nie jest poprawne w środowisku klienta MQI produktu MQ . Próba użycia wywołania nie powiodła się z kodem przyczyny MQRC_ENVIRONMENT_ERROR.
5. Jeśli menedżer kolejek działa jako koordynator jednostki pracy dla globalnych jednostek pracy, menedżery zasobów, które mogą uczestniczyć w jednostce pracy, są zdefiniowane w pliku konfiguracyjnym menedżera kolejek.
6. W systemie IBM iobsługiwane są trzy typy jednostek pracy:
   - Opcja **Koordynowana przez menedżera kolejek lokalna jednostka pracy** może być używana tylko wtedy, gdy definicja kontroli transakcji nie istnieje na poziomie zadania, tzn. komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie może być wydana dla zadania.
   - **Globalna jednostka pracy koordynowana przez menedżer kolejek** nie jest obsługiwana.
   - **Globalna jednostka pracy koordynowana zewnętrznie** może być używana tylko wtedy, gdy definicja kontroli transakcji istnieje na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** musi zostać wydana dla zadania. Jeśli zostało to zrobione, operacje IBM i COMMIT i ROLLBACK mają zastosowanie do zasobów produktu MQ , a także do zasobów należących do innych uczestniczących menedżerów zasobów.

Wywołanie C

```
MQBEGIN (Hconn, &BeginOptions, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;         /* Connection handle */
MQBO     BeginOptions;  /* Options that control the action of MQBEGIN */
```

```
MQLONG   CompCode;      /* Completion code */
MQLONG   Reason;        /* Reason code qualifying CompCode */
```

Wywołanie COBOL

```
CALL 'MQBEGIN' USING HCONN, BEGINOPTIONS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQBEGIN (Hconn, BeginOptions, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie Visual Basic

```
MQBEGIN Hconn, BeginOptions, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-Przekształć bufor w uchwyt komunikatu

Wywołanie funkcji MQBUFMH przekształca bufor w uchwyt komunikatu i jest odwrotnością wywołania MQMHBUF.

To wywołanie pobiera deskryptor komunikatu i właściwości MQRFH2 w buforze i udostępnia je za pośrednictwem uchwytu komunikatu. Właściwości MQRFH2 w danych komunikatu są opcjonalnie usuwane. Pola *Encoding*, *CodedCharSetId*i *Format* deskryptora komunikatu są aktualizowane, jeśli jest to konieczne, w celu poprawnego opisania zawartości buforu po usunięciu właściwości.

Składnia

MQBUFMH (*Hconn*, *Hmsg*, *BufMsgHOpts*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *Compcode*, *Reason*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość **Hconn** musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony za pomocą wywołania MQHC_UNASSOCIATED_HCONN, należy nawiązać poprawne połączenie w wątku, przekształcając bufor w uchwyt komunikatu. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie nie powiedzie się i zostanie zwrócony komunikat MQRC_CONNECTION_BROKEN.

Komunikat Hmsg
Typ: MQHMQSG-input

Jest to uchwyt komunikatu, dla którego wymagany jest bufor. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

BufMsgHOpts
Typ: MQBMHO-wejście

Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia uchwytów komunikatów z buforów.

Szczegółowe informacje można znaleźć w sekcji “MQBMHO-Opcje obsługi bufor-komunikat” na stronie 278.

MsgDesc
Typ: MQMD-wejście/wyjście

Struktura *MsgDesc* zawiera właściwości deskryptora komunikatu i opisuje treść obszaru buforu.

W danych wyjściowych wywołania właściwości są opcjonalnie usuwane z obszaru buforu i w tym przypadku deskryptor komunikatu jest aktualizowany w celu poprawnego opisania obszaru buforu.

Dane w tej strukturze muszą być w zestawie znaków i kodowaniu aplikacji.

BufferLength
Typ: MQLONG-input

BufferLength to długość obszaru buforu w bajtach.

Wartość *BufferLength* równa zero bajtów jest poprawna i wskazuje, że obszar buforu nie zawiera danych.

Buforuj
Typ: MQBYTExBufferLength-input/output

Są to opcje, które sterują działaniem komendy MQBEGIN zgodnie z opisem w sekcji “MQBEGIN-Rozpoczęcie jednostki pracy” na stronie 651.

Buffer definiuje obszar zawierający bufor komunikatów. W przypadku większości danych należy wyrównać bufor do granicy 4-bajtowej.

Jeśli plik **Buffer** zawiera dane znakowe lub liczbowe, należy ustawić pola *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** na wartości odpowiednie dla danych. W razie potrzeby umożliwia to konwersję danych.

Jeśli w buforze komunikatów zostaną znalezione właściwości, zostaną one opcjonalnie usunięte. Później staną się dostępne w uchwycie komunikatu po powrocie z wywołania.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void, co oznacza, że jako parametr można podać adres dowolnego typu danych.

Jeśli parametr **BufferLength** ma wartość zero, nie jest przywoływany parametr **Buffer** ; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość null.

DataLength
Typ: MQLONG-wyjście

Długość (w bajtach) buforu, z którego mogły zostać usunięte właściwości.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

BŁĄD MQRC_BMHO_ERROR
(2489, X'09B9') Niepoprawna struktura opcji uchwytu buforu do komunikatu.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'07D4') Niepoprawny parametr buforu.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

MQRC_MD_BŁĄD
(2026, X'07EA') Niepoprawny deskryptor komunikatu.

MQRC_MSG_HANDLE_IN_USE,
(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_RFH_ERROR
(2334, X'091E') Niepoprawna struktura MQRFH2 .

BŁĄD MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

Wywołania MQBUFMH nie mogą być przechwytywane przez wyjścia API-bufor jest przekształcany w uchwyt komunikatu w obszarze aplikacji; wywołanie nie dociera do menedżera kolejek.

Wywołanie C

```
MQBUFMH (Hconn, Hmsg, &BufMsgHOpts, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQBUFMH' USING HCONN, HMSG, BUFMSGHOPTS, MSGDESC, BUFFERLENGTH,
                           BUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQBUFMH (Hconn, Hmsg, BufMsgHOpts, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQBUFMH,(HCONN,HMSG,BUFMSGHOPTS,MSGDESC,BUFFERLENGTH,BUFFER,
               DATALENGTH,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-zarządzanie wywołaniem zwrotnym

Wywołanie MQCB rejestruje wywołanie zwrotne dla określonego uchwytu obiektu i steruje aktywowaniem i zmianami wywołania zwrotnego.

Wywołanie zwrotne jest fragmentem kodu (określonym jako nazwa funkcji, która może być połączona dynamicznie lub jako wskaźnik funkcji), który jest wywoływany przez funkcję IBM MQ w przypadku wystąpienia określonych zdarzeń.

Aby można było używać obiektów MQCB i MQCTL na kliencie, należy nawiązać połączenie z serwerem, na którym wynegocjowany parametr **SHARECNV** kanału uzgodnił wartość niezerową.

Można zdefiniować następujące typy wywołań zwrotnych:

Konsument komunikatu
Funkcja zwrotna konsumenta komunikatów jest wywoływana, gdy komunikat spełniający określone kryteria wyboru jest dostępny w uchwycie obiektu.

Dla każdego uchwytu obiektu można zarejestrować tylko jedną funkcję zwrotną. Jeśli jedna kolejka ma być odczytywana z wieloma kryteriami wyboru, kolejka musi być otwierana wiele razy, a funkcja konsumenta musi być zarejestrowana w każdym uchwycie.

procedura obsługi zdarzeń
Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko wywołania zwrotnego.

Funkcja jest wywoływana w przypadku wystąpienia warunku zdarzenia, na przykład zatrzymania lub wyciszenia menedżera kolejek lub połączenia.

Funkcja nie jest wywoływana dla warunków specyficznych dla pojedynczego konsumenta komunikatów, na przykład MQRC_GET_INHIBITED; jest wywoływana, jeśli funkcja zwrotna nie zakończy się normalnie.

Składnia

MQCB (*Hconn*, *Operacja*, *CallbackDesc*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *CompCode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można określić następującą wartość specjalną dla parametru *MQHC_DEF_HCONN* , aby użyć uchwytu połączenia powiązanego z tą jednostką wykonywania.

Operacja
Typ: MQLONG-input

Operacja przetwarzana w wywołaniu zwrotnym zdefiniowanym dla określonego uchwytu obiektu. Należy podać jedną z następujących opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

MQOP_REGISTER,
Zdefiniuj funkcję zwrotną dla określonego uchwytu obiektu. Ta operacja definiuje funkcję, która ma zostać wywołana, oraz kryteria wyboru, które mają zostać użyte.

Jeśli funkcja zwrotna jest już zdefiniowana dla uchwytu obiektu, definicja jest zastępowana. Jeśli podczas zastępowania wywołania zwrotnego zostanie wykryty błąd, funkcja zostanie wyrejestrowana.

Jeśli wywołanie zwrotne jest zarejestrowane w tej samej funkcji zwrotnej, w której zostało wcześniej wyrejestrowane, jest ono traktowane jako operacja zastąpienia; żadne wywołania początkowe lub końcowe nie są wywoływane.

Opcji MQOP_REGISTER można używać z opcją MQOP_SUSPEND lub MQOP_RESUME.

MQOP_DEREGISTER
Zatrzymaj odbieranie komunikatów dla uchwytu obiektu i usuń uchwyt z tych, które zostały zakwalifikowane do wywołania zwrotnego.

Wywołanie zwrotne jest automatycznie wyrejestrowywane, jeśli powiązany uchwyt jest zamknięty.

Jeśli wywołanie MQOP_DEREGISTER jest wykonywane z poziomu konsumenta, a wywołanie zwrotne ma zdefiniowane wywołanie zatrzymania, jest ono wywoływane po powrocie z konsumenta.

Jeśli ta operacja jest wykonywana dla *Hobj* bez zarejestrowanego konsumenta, wywołanie zwraca wartość MQRC_CALLBACK_NOT_REGISTERED.

MQOP_ZAWIEŚ
Zawiesza odbieranie komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie mieć zdarzeń zawieszonych, a wszystkie zdarzenia pominięte w stanie zawieszenia nie będą udostępniane operacji po jej wznowieniu.

Po zawieszeniu funkcja konsumenta kontynuuje pobieranie wywołań zwrotnych typu sterującego.

MQOP_RESUME
Wznawia odbieranie komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie mieć zdarzeń zawieszonych, a wszystkie zdarzenia pominięte w stanie zawieszenia nie będą udostępniane operacji po jej wznowieniu.

CallbackDesc
Typ: MQCBD-wejście

Jest to struktura identyfikująca funkcję zwrotną, która jest rejestrowana przez aplikację, oraz opcje używane podczas jej rejestrowania.

Szczegółowe informacje na temat struktury zawiera sekcja MQCBD .

Deskryptor wywołania zwrotnego jest wymagany tylko w przypadku opcji MQOP_REGISTER. Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

Hobj
Typ: MQHOBJ-input

Ten uchwyt reprezentuje dostęp, który został ustanowiony dla obiektu, z którego ma być pobierany komunikat. Jest to uchwyt zwrócony z poprzedniego wywołania MQOPEN lub MQSUB (w parametrze **Hobj**).

Parametr *Hobj* nie jest wymagany podczas definiowania procedury obsługi zdarzeń (MQCBT_EVENT_HANDLER) i powinien być określony jako MQHO_NONE.

Jeśli funkcja *Hobj* została zwrócona z wywołania MQOPEN, kolejka musi być otwarta z co najmniej jedną z następujących opcji:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE (wyłączna)
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE,

MsgDesc
Typ: MQMD-input

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu.

Parametr **MsgDesc** definiuje atrybuty komunikatów wymaganych przez konsumenta oraz wersję deskryptora MQMD przekazywanego do konsumenta komunikatów.

Do wyboru komunikatów używane są parametry *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*i *Offset* w strukturze MQMD, w zależności od opcji określonych w parametrze **GetMsgOpts** .

Jeśli określono opcję MQGMO_CONVERT, do konwersji komunikatów używane są parametry *Encoding* i *CodedCharSetId* .

Szczegółowe informacje na ten temat zawiera sekcja MQMD .

Parametr *MsgDesc* jest używany dla tabeli MQOP_REGISTER, jeśli wymagane są wartości inne niż domyślne dla dowolnych pól. Wartość *MsgDesc* nie jest używana dla procedury obsługi zdarzeń.

Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

Należy zauważyć, że jeśli wiele konsumentów jest zarejestrowanych dla tej samej kolejki z nakładającymi się selektorami, wybrany konsument dla każdego komunikatu jest niezdefiniowany.

GetMsgOpcje
Typ: MQGMO-wejście

Parametr **GetMsgOpts** steruje sposobem, w jaki konsument komunikatów otrzymuje komunikaty. Wszystkie opcje tego parametru mają znaczenie opisane w sekcji “MQGMO-opcje pobierania komunikatów” na stronie 376, jeśli są używane w wywołaniu MQGET, z wyjątkiem następujących:

MQGMO_SET_SIGNAL
Ta opcja nie jest dozwolona.

MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT, MQGMO_MARK_ *
Kolejność komunikatów dostarczanych do konsumenta przeglądania zależy od kombinacji tych opcji. Istotne kombinacje to:

MQGMO_BROWSE_FIRST
Pierwszy komunikat w kolejce jest wielokrotnie dostarczany do konsumenta. Jest to przydatne, gdy konsument niszczy komunikat w wywołaniu zwrotnym. Tej opcji należy używać ostrożnie.

MQGMO_BROWSE_NEXT
Konsument otrzymuje każdy komunikat w kolejce, od bieżącej pozycji kursora do osiągnięcia końca kolejki.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT
Kursor zostanie zresetowany do początku kolejki. Konsumentowi są następnie nadawane wszystkie komunikaty, dopóki kursor nie osiągnie końca kolejki.

MQGMO_BROWSE_FIRST + MQGMO_MARK_ *
Począwszy od początku kolejki, konsumentowi jest nadawany pierwszy nieoznaczony komunikat w kolejce, który jest następnie oznaczany dla tego konsumenta. Ta kombinacja zapewnia, że konsument może odbierać nowe komunikaty dodane za bieżącym punktem kursora.

MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
Począwszy od pozycji kursora, konsument otrzymuje następny nieoznaczony komunikat w kolejce, który jest następnie oznaczany dla tego konsumenta. Tej kombinacji należy używać ostrożnie, ponieważ komunikaty mogą być dodawane do kolejki za bieżącą pozycją kursora.

MQGMO_BROWSE_FIRST + MQGMO_BROWSE_NEXT + MQGMO_MARK_ *
Ta kombinacja nie jest dozwolona. W przypadku użycia wywołania zwraca MQRC_OPTIONS_ERROR.

MQGMO_NO_WAIT, MQGMO_WAIT i WaitInterval
Te opcje sterują sposobem wywoływania konsumenta.

MQGMO_NO_WAIT
Konsument nie jest nigdy wywoływany z opcją MQRC_NO_MSG_AVAILABLE. Konsument jest wywoływany tylko dla komunikatów i zdarzeń.

MQGMO_WAIT z zerową wartością WaitInterval
Kod MQRC_NO_MSG_AVAILABLE jest przekazywany do konsumenta, gdy nie ma dostępnych komunikatów, a konsument został uruchomiony lub został dostarczony co najmniej jeden komunikat od ostatniego kodu przyczyny "brak komunikatów".

Zapobiega to odpytywaniu konsumenta w pętli zajętości, gdy określony jest zerowy odstęp czasu oczekiwania.

MQGMO_WAIT i dodatni WaitInterval
Konsument jest wywoływany po upływie określonego czasu oczekiwania z kodem przyczyny MQRC_NO_MSG_AVAILABLE. To wywołanie jest wykonywane niezależnie od tego, czy jakiekolwiek komunikaty zostały dostarczone do konsumenta. Pozwala to użytkownikowi na wykonywanie przetwarzania pulsu lub przetwarzania wsadowego.

MQGMO_WAIT i WaitInterval dla MQWI_UNLIMITED
Określa nieskończony czas oczekiwania przed zwróceniem komunikatu MQRC_NO_MSG_AVAILABLE. Konsument nie jest nigdy wywoływany z opcją MQRC_NO_MSG_AVAILABLE.

Parametr *GetMsgOpts* jest używany tylko w przypadku opcji MQOP_REGISTER i wtedy, gdy wymagane są wartości inne niż domyślne dla dowolnych pól. Wartość *GetMsgOpts* nie jest używana dla procedury obsługi zdarzeń.

Jeśli parametr *GetMsgOpts* nie jest wymagany, przekazany adres parametru może mieć wartość NULL. Użycie tego parametru jest równoważne określeniu opcji MQGMO_DEFAULT razem z opcją MQGMO_FAIL_IF_QUIESCING.

Jeśli uchwyt właściwości komunikatu jest udostępniany w strukturze MQGMO, kopia jest udostępniana w strukturze MQGMO, która jest przekazywana do wywołania zwrotnego konsumenta. Po powrocie z wywołania MQCB aplikacja może usunąć uchwyt właściwości komunikatu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Na poniższej liście znajdują się kody przyczyny, które menedżer kolejek może zwrócić dla parametru **Reason** .

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

BŁĄD MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') Niepoprawne pole typu wywołania zwrotnego.

MQRC_CALLBACK_NOT_REGISTERED
(2448, X' 990 ') Nie można wyrejestrować, zawiesić lub wznowić, ponieważ nie ma zarejestrowanego wywołania zwrotnego.

BŁĄD MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') Należy podać wartość *CallbackFunction* lub *CallbackName* , ale nie obie jednocześnie.

BŁĄD MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Niepoprawne pole typu wywołania zwrotnego.

BŁĄD MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Niepoprawne pole opcji MQCBD.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CORREL_ID_ERROR
(2207, X'89F') Błąd identyfikatora korelacji.

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Niepoprawny parametr długości danych.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

MQRC_GET_INHIBITED
(2016, X'7E0') Liczba pobrań zablokowana dla kolejki.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Globalny konflikt jednostek pracy.

BŁĄD MQRC_GMO_ERROR
(2186, X'88A') Niepoprawna struktura opcji Get-message.

MQRC_HANDLE_IN_USE_FOR_UOW,
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_INCONSISTENT_BROWSE (przeglądanie niespójne)
(2259, X'8D3') Niespójna specyfikacja przeglądania.

MQRC_INCONSISTENT_UOW,
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_INVALID_MSG_UNDER_CURSOR,
(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

MQRC_LOCAL_UOW_CONFLICT (konflikt uow_rejestru)
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

BŁĄD MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Opcje zgodności są niepoprawne.

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B4') Niepoprawne pole *MaxMsgLength* .

MQRC_MD_BŁĄD
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') W module nie można znaleźć określonego punktu wejścia funkcji.

MQRC_MODULE_INVALID
(2496, X'9C0') Znaleziono moduł, ale ma on niepoprawny typ; nie jest to 32-bitowa, 64-bitowa lub poprawna biblioteka dołączana dynamicznie.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie ma uprawnień do ładowania.

BŁĄD MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

MQRC_MSG_TOKEN_ERROR,
(2331, X'91B') Użycie znacznika komunikatu jest niepoprawne.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') Brak dostępnego komunikatu.

MQRC_NO_MSG_UNDER_CURSOR,
(2034, X'7F2') Kursor przeglądania nie jest ustawiony na komunikacie.

MQRC_NOT_OPEN_FOR_BROWSE,
(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

MQRC_NOT_OPEN_FOR_INPUT (MQRC_NOT_OPEN_PUT)
(2037, X'7F5') Kolejka nie jest otwarta do wprowadzania.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

BŁĄD OPERACJI MQRC
(2206, X'89E') Niepoprawny kod operacji w wywołaniu API.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

BŁĄD MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_SIGNAL_OCZEKUJĄCE
(2069, X'815 ') Sygnał zaległości dla tego uchwytu.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

MQRC_SYNCPOINT_NIEDOSTĘPNE
(2072, X'818 ') Obsługa punktu synchronizacji jest niedostępna.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

MQRC_UOW_NIEDOSTĘPNE
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

BŁĄD INTERVAL_MQRC_WAIT_ERROR
(2090, X'82A') Niepoprawny odstęp czasu oczekiwania w MQGMO.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Podano niewłaściwą wersję MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Obiekt MQCB jest używany do definiowania działania, które ma być wywoływane dla każdego komunikatu, zgodnego z określonymi kryteriami, dostępnego w kolejce. Podczas przetwarzania działania komunikat jest usuwany z kolejki i przekazywany do zdefiniowanego konsumenta komunikatów lub udostępniany jest znacznik komunikatu, który jest używany do pobierania komunikatu.
2. Obiekt MQCB może być używany do definiowania podprogramów wywołania zwrotnego przed rozpoczęciem korzystania z obiektu MQCTL lub może być używany z poziomu podprogramu wywołania zwrotnego.
3. Aby użyć obiektu MQCB spoza procedury wywołania zwrotnego, należy najpierw zawiesić przetwarzanie komunikatów przy użyciu obiektu MQCTL i wznowić przetwarzanie.
4. Baza MQCB nie jest obsługiwana w adapterze IMS .

Sekwencja wywołań zwrotnych konsumenta komunikatów

Konsument można skonfigurować w taki sposób, aby wywoływał wywołanie zwrotne w punktach kluczowych podczas cyklu życia konsumenta. Na przykład:

- gdy konsument jest po raz pierwszy zarejestrowany,
- po uruchomieniu połączenia,
- po zatrzymaniu połączenia i
- gdy konsument jest wyrejestrowany jawnie lub niejawnie przez MQCLOSE.

Tabela 541. Definicje komend MQCTL

Czasownik	**Znaczenie**
MQCTL (URUCHOM)	Wywołanie MQCTL przy użyciu operacji MQOP_START
MQCTL (ZATRZYMAJ)	Wywołanie MQCTL przy użyciu operacji MQOP_STOP
MQCTL (OCZEKIWANIE)	Wywołanie MQCTL przy użyciu operacji MQOP_START_WAIT

Umożliwia to konsumentowi zachowanie stanu powiązanego z konsumentem. Gdy aplikacja żąda wywołania zwrotnego, reguły wywołania konsumenta są następujące:

ZAREJESTRUJ SIĘ
Jest zawsze pierwszym typem wywołania wywołania zwrotnego.

Jest zawsze wywoływana w tym samym wątku, co wywołanie MQCB (REGISTER).

START
Jest zawsze wywoływana synchronicznie z komendą MQCTL (START).

- Wszystkie wywołania zwrotne komendy START są wykonywane przed zwróceniem komendy MQCTL (START).

jest w tym samym wątku, co dostarczanie komunikatów, jeśli zażądano THREAD_AFFINITY.

Wywołanie z komendą start nie jest gwarantowane, jeśli na przykład poprzednie wywołanie zwrotne wywołuje komendę MQCTL (STOP) podczas wykonywania komendy MQCTL (START).

STOP
Kolejne komunikaty i zdarzenia nie będą dostarczane po tym wywołaniu, dopóki połączenie nie zostanie zrestartowane.

Komenda STOP jest gwarantowana, jeśli aplikacja została wcześniej wywołana dla komendy START, komunikatu lub zdarzenia.

DEREGISTER (Deregator)
Jest zawsze ostatnim typem wywołania wywołania zwrotnego.

Upewnij się, że aplikacja wykonuje inicjowanie i czyszczenie w oparciu o wątki w wywołaniach zwrotnych START i STOP. Za pomocą wywołań zwrotnych REGISTER i DEREGISTER można wykonywać inicjację i procedurę czyszczącą nie opartą na wątkach.

Nie należy zakładać, że czas życia i dostępność wątku jest inna niż to, co zostało określone. Na przykład nie należy polegać na wątku, który pozostaje aktywny po ostatnim wywołaniu funkcji DEREGISTER. Podobnie, jeśli nie wybrano opcji THREAD_AFFINITY, nie należy zakładać, że wątek istnieje przy każdym uruchomieniu połączenia.

Jeśli aplikacja ma określone wymagania dotyczące parametrów wątków, zawsze może utworzyć odpowiedni wątek, a następnie użyć komendy MQCTL (WAIT). Spowoduje to przekazanie wątku do IBM MQ w celu asynchronicznego dostarczania komunikatów.

Użycie połączenia konsumenta komunikatów

Konsument można skonfigurować w taki sposób, aby wywoływał wywołanie zwrotne w punktach kluczowych podczas cyklu życia konsumenta. Na przykład:

- gdy konsument jest po raz pierwszy zarejestrowany,
- po uruchomieniu połączenia,
- po zatrzymaniu połączenia i
- gdy konsument jest wyrejestrowany jawnie lub niejawnie przez MQCLOSE.

Tabela 542. Definicje komend MQCTL

Czasownik	**Znaczenie**
MQCTL (URUCHOM)	Wywołanie MQCTL przy użyciu operacji MQOP_START
MQCTL (ZATRZYMAJ)	Wywołanie MQCTL przy użyciu operacji MQOP_STOP
MQCTL (OCZEKIWANIE)	Wywołanie MQCTL przy użyciu operacji MQOP_START_WAIT

Umożliwia to konsumentowi zachowanie stanu powiązanego z konsumentem. Gdy aplikacja żąda wywołania zwrotnego, reguły wywołania konsumenta są następujące:

ZAREJESTRUJ SIĘ
Jest zawsze pierwszym typem wywołania wywołania zwrotnego.

Jest zawsze wywoływana w tym samym wątku, co wywołanie MQCB (REGISTER).

START
Jest zawsze wywoływana synchronicznie z komendą MQCTL (START).

- Wszystkie wywołania zwrotne komendy START są wykonywane przed zwróceniem komendy MQCTL (START).

jest w tym samym wątku, co dostarczanie komunikatów, jeśli zażądano THREAD_AFFINITY.

Wywołanie z komendą start nie jest gwarantowane, jeśli na przykład poprzednie wywołanie zwrotne wywołuje komendę MQCTL (STOP) podczas wykonywania komendy MQCTL (START).

STOP
Kolejne komunikaty i zdarzenia nie będą dostarczane po tym wywołaniu, dopóki połączenie nie zostanie zrestartowane.

Komenda STOP jest gwarantowana, jeśli aplikacja została wcześniej wywołana dla komendy START, komunikatu lub zdarzenia.

DEREGISTER (Deregator)
Jest zawsze ostatnim typem wywołania wywołania zwrotnego.

Upewnij się, że aplikacja wykonuje inicjowanie i czyszczenie w oparciu o wątki w wywołaniach zwrotnych START i STOP. Za pomocą wywołań zwrotnych REGISTER i DEREGISTER można wykonywać inicjację i procedurę czyszczącą nie opartą na wątkach.

Nie należy zakładać, że czas życia i dostępność wątku jest inna niż to, co zostało określone. Na przykład nie należy polegać na wątku, który pozostaje aktywny po ostatnim wywołaniu funkcji DEREGISTER. Podobnie, jeśli nie wybrano opcji THREAD_AFFINITY, nie należy zakładać, że wątek istnieje przy każdym uruchomieniu połączenia.

Jeśli aplikacja ma określone wymagania dotyczące parametrów wątków, zawsze może utworzyć odpowiedni wątek, a następnie użyć komendy MQCTL (WAIT). Spowoduje to przekazanie wątku do IBM MQ w celu asynchronicznego dostarczania komunikatów.

Wywołanie C

```
MQCB (Hconn, Operation, CallbackDesc, Hobj, MsgDesc,
GetMsgOpts, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQCB' USING HCONN, OPERATION, CBDESC, HOBJ, MSGDESC,
           GETMSGOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQCB(Hconn, Operation, CallbackDesc, Hobj, MsgDesc, GetMsgOpts,
          CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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-funkcja wywołania zwrotnego

Wywołanie funkcji MQCB_FUNCTION jest funkcją zwrotną do obsługi zdarzeń i asynchronicznego korzystania z komunikatów.

Definicja wywołania MQCB_FUNCTION jest udostępniana wyłącznie w celu opisania parametrów przekazywanych do funkcji zwrotnej. Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQCB_FUNCTION.

Specyfikacja funkcji, która ma zostać wywołana, jest wejściem do wywołania MQCB i jest przekazywana za pośrednictwem struktury MQCBD .

Składnia

MQCB_FUNCTION (*Hconn*, *MsgDesc*, *GetMsgOpts*, *Buffer*, *Context*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX. W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i określić następującą wartość dla Hconn:

MQHC_DEF_KONN

Domyślny uchwyt połączenia.

MsgDesc
Typ: MQMD-input

Ta struktura opisuje atrybuty pobranego komunikatu.

Szczegółowe informacje można znaleźć w sekcji “MQMD-deskryptor komunikatu” na stronie 432.

Wersja przekazanego deskryptora MQMD jest taka sama, jak wersja przekazana w wywołaniu MQCB, w którym zdefiniowano funkcję konsumenta.

Adres deskryptora MQMD jest przekazywany jako znaki o wartości NULL, jeśli do żądania zwrócenia uchwytu komunikatu zamiast deskryptora MQMD użyto obiektu MQGMO w wersji 4.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

Opcja GetMsg
Typ: MQGMO-wejście

Opcje używane do sterowania działaniami konsumenta komunikatów. Ten parametr zawiera również dodatkowe informacje o zwróconym komunikacie.

Szczegółowe informacje na ten temat zawiera sekcja MQGMO .

Przekazana wersja MQGMO jest najnowszą obsługiwaną wersją.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

Buforuj
Typ: MQBYTExBuffer-długość-wejście

Jest to obszar zawierający dane komunikatu.

Jeśli dla tego wywołania nie jest dostępny żaden komunikat lub jeśli komunikat nie zawiera danych komunikatu, adres *Buffer* jest przekazywany jako wartości null.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

Kontekst
Typ: MQCBC-input/output

Ta struktura udostępnia informacje o kontekście funkcjom zwrotnym. Szczegółowe informacje można znaleźć w sekcji “MQCBC-kontekst wywołania zwrotnego” na stronie 285.

Użycie notatek

1. Należy pamiętać, że jeśli procedury zwrotne używają usług, które mogą opóźnić lub zablokować wątek, na przykład MQGET z oczekiwaniem, może opóźnić rozsyłanie innych wywołań zwrotnych.
2. Oddzielna jednostka pracy nie jest automatycznie ustanawiana dla każdego wywołania procedury zwrotnej, dlatego procedury mogą albo wydać wywołanie zatwierdzenia, albo odroczyć zatwierdzenie, aż do przetworzenia logicznego zadania wsadowego. Po zatwierdzeniu zadania wsadowego zatwierdza on komunikaty dla wszystkich funkcji zwrotnych, które zostały wywołane od ostatniego punktu synchronizacji.
3. Programy wywoływane przez komendę CICS LINK lub CICS START pobierają parametry za pomocą usług CICS za pośrednictwem nazwanych obiektów nazywanych kontenerami kanałów. Nazwy kontenerów są takie same, jak nazwy parametrów. Więcej informacji na ten temat zawiera dokumentacja systemu CICS .
4. Procedury zwrotne mogą wywoływać wywołania MQDISC, ale nie dla własnego połączenia. Jeśli na przykład procedura zwrotna utworzyła połączenie, może również rozłączyć połączenie.
5. Procedura zwrotna nie powinna generalnie polegać na każdym wywołaniu z tego samego wątku. Jeśli jest to wymagane, należy użyć parametru MQCTLO_THREAD_AFFINITY podczas uruchamiania połączenia.
6. Gdy procedura zwrotna otrzyma niezerowy kod przyczyny, musi podjąć odpowiednie działanie.
7. Opcja MQCB_FUNCTION nie jest obsługiwana w adapterze IMS .

MQCLOSE-zamknięcie obiektu

Wywołanie MQCLOSE powoduje ponowne uzyskanie dostępu do obiektu i jest odwrotnością wywołań MQOPEN i MQSUB.

Składnia

MQCLOSE (*Hconn*, *Hobj*, *Opcje*, *CompCode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie z/OS dla aplikacji CICS można pominąć wywołanie MQCONN i określić następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Hobj
Typ: MQHOBJ-input/output

Ten uchwyt reprezentuje obiekt, który jest zamykany. Obiekt może być dowolnego typu. Wartość *Hobj* została zwrócona przez poprzednie wywołanie MQOPEN.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia ten parametr na wartość, która nie jest poprawnym uchwytem dla środowiska. Wartość ta jest następująca:

MQHO_UNUSABLE_HOBJ
Uchwyt obiektu nie do użycia.

W systemie z/OSparametr *Hobj* jest ustawiany na wartość, która nie jest zdefiniowana.

Opcje
Typ: MQLONG-input

Ten parametr steruje sposobem zamykania obiektu.

Tylko trwałe kolejki dynamiczne i subskrypcje mogą być zamykane w więcej niż jeden sposób, ponieważ muszą być zachowane lub usunięte. Są to kolejki z atrybutem **DefinitionType** , który ma wartość MQQDT_PERMANENT_DYNAMIC (patrz opis atrybutu **DefinitionType** w sekcji "Atrybuty kolejek" na stronie 863). W tym temacie przedstawiono podsumowanie opcji zamykania.

Trwałe subskrypcje można zachować lub usunąć. Są one tworzone za pomocą wywołania MQSUB z opcją MQSO_DURABLE.

Podczas zamykania uchwytu do zarządzanego miejsca docelowego (czyli parametru **Hobj** zwróconego w wywołaniu MQSUB używającym opcji MQSO_MANAGED) menedżer kolejek czyści wszystkie publikacje, które nie zostały pobrane po usunięciu powiązanej subskrypcji. Subskrypcja jest usuwana przy użyciu opcji MQCO_REMOVE_SUB parametru **Hsub** zwróconego w wywołaniu MQSUB. Należy zauważyć, że opcja MQCO_REMOVE_SUB jest domyślnym zachowaniem programu MQCLOSE w przypadku subskrypcji nietrwałej.

Podczas zamykania uchwytu do niezarządzanego miejsca docelowego użytkownik jest odpowiedzialny za czyszczenie kolejki, do której są wysyłane publikacje. Najpierw zamknij subskrypcję za pomocą komendy MQCO_REMOVE_SUB, a następnie przetwórz komunikaty poza kolejką, aż nie pozostaną żadne komunikaty.

Należy podać tylko jedną opcję z następujących:

Opcje kolejki dynamicznej: Te opcje sterują sposobem zamykania trwałych kolejek dynamicznych.

MQCO_DELETE
Kolejka jest usuwana, jeśli spełniony jest jeden z następujących warunków:

- Jest to trwała kolejka dynamiczna, utworzona przez poprzednie wywołanie MQOPEN i nie ma w niej żadnych komunikatów ani niezatwierdzonych żądań pobierania lub umieszczania oczekujących dla kolejki (dla bieżącego zadania lub dowolnego innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło kod *Hobj*. W takim przypadku wszystkie komunikaty w kolejce są czyszczone.

We wszystkich innych przypadkach, w tym w przypadku zwrócenia wartości *Hobj* w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OPTION_NOT_VALID_FOR_TYPE i obiekt nie jest usuwany.

W systemie z/OS, jeśli kolejka jest kolejką dynamiczną, która została logicznie usunięta i jest to jej ostatni uchwyt, kolejka jest fizycznie usuwana. Więcej informacji na ten temat zawiera sekcja “Użycie notatek” na stronie 675.

MQCO_DELETE_PURGE
Kolejka zostanie usunięta, a wszystkie znajdujące się w niej komunikaty zostaną wyczyszczone, jeśli spełniony jest jeden z następujących warunków:

- Jest to trwała kolejka dynamiczna, utworzona przez poprzednie wywołanie MQOPEN i nie ma żadnych niezatwierdzonych żądań pobierania lub umieszczania oczekujących dla kolejki (dla bieżącego zadania lub dowolnego innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło kod *Hobj*.

We wszystkich innych przypadkach, w tym w przypadku zwrócenia wartości *Hobj* w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OPTION_NOT_VALID_FOR_TYPE i obiekt nie jest usuwany.

Tabela 543. Opcje zamykania dla różnych typów obiektów

Typ obiektu lub kolejki	**MQCO_BRAK**	**MQCO_DELETE**	**MQCO_DELETE_PURGE**
Obiekt inny niż kolejka	Zachowany	Niepoprawne	Niepoprawne
Predefiniowana kolejka	Zachowany	Niepoprawne	Niepoprawne
stała kolejka dynamiczna	Zachowany	Usunięte, jeśli puste i bez oczekujących aktualizacji	Komunikaty usunięte; kolejka usunięta, jeśli nie ma oczekujących aktualizacji
Tymczasowa kolejka dynamiczna (wywołanie wysłane przez twórcę kolejki)	Usunięte	Usunięte	Usunięte
Tymczasowa kolejka dynamiczna (wywołanie nie zostało wykonane przez twórcę kolejki)	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna	Zachowany	Niepoprawne	Niepoprawne
Miejsce docelowe subskrypcji zarządzanej	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna (subskrypcja została usunięta)	Komunikaty usunięte; kolejka usunięta	Niepoprawne	Niepoprawne

Opcje zamykania subskrypcji: Te opcje określają, czy trwałe subskrypcje są usuwane po zamknięciu uchwytu oraz czy publikacje oczekujące na odczyt przez aplikację są czyszczone. Te opcje są poprawne tylko w przypadku użycia z uchwytem obiektu zwróconym w parametrze **Hsub** wywołania MQSUB.

MQCO_KEEP_SUB
Uchwyt subskrypcji jest zamknięty, ale wykonana subskrypcja jest zachowana. Publikacje będą nadal wysyłane do miejsca docelowego określonego w subskrypcji. Ta opcja jest poprawna tylko wtedy, gdy subskrypcja została wykonana z opcją MQSO_DURABLE.

MQCO_KEEP_SUB jest wartością domyślną, jeśli subskrypcja jest trwała

MQCO_REMOVE_SUB
Subskrypcja zostanie usunięta, a uchwyt do subskrypcji zostanie zamknięty.

Parametr **Hobj** wywołania MQSUB nie jest unieważniany przez zamknięcie parametru **Hsub** i może być nadal używany do odbierania pozostałych publikacji przez wywołania MQGET lub MQCB. Po zamknięciu parametru **Hobj** wywołania MQSUB, jeśli było to zarządzane miejsce docelowe, usuwane są wszystkie niepobrane publikacje.

MQCO_REMOVE_SUB jest wartością domyślną, jeśli subskrypcja nie jest trwała.

Pomyślne zakończenie operacji MQCO_REMOVE_SUB nie oznacza, że działanie zostało zakończone. Aby sprawdzić, czy wywołanie zostało zakończone, zapoznaj się z krokiem DELETE SUB w sekcji Sprawdzanie, czy komendy asynchroniczne dla sieci rozproszonych zostały zakończone.

Te opcje zamykania subskrypcji zostały podsumowane w poniższych tabelach.

Tabela 544. Opcje zamykania uchwytu trwałej subskrypcji, ale zachowywanie subskrypcji

Czynność	**Opcja zamknięcia subskrypcji**
Zachowaj publikacje w uchwycie MQOPENed	MQCO_KEEP_SUB
Usuń publikacje z uchwytu MQOPENed	Działanie niedozwolone
Zachowaj publikacje w uchwycie MQSO_MANAGED	MQCO_KEEP_SUB
Usuwanie publikacji z uchwytu MQSO_MANAGED	Działanie niedozwolone

Aby anulować subskrypcję, zamykając uchwyt subskrypcji trwałej i anulując jego subskrypcję lub zamykając uchwyt subskrypcji nietrwałej, należy użyć następujących opcji zamykania subskrypcji:

Tabela 545. Opcje anulowania subskrypcji

Czynność	**Opcja zamknięcia subskrypcji**
Zachowaj publikacje w uchwycie MQOPENed	MQCO_REMOVE_SUB
Usuń publikacje z uchwytu MQOPENed	Działanie niedozwolone
Zachowaj publikacje w uchwycie MQSO_MANAGED	MQCO_REMOVE_SUB

Opcje odczytu z wyprzedzeniem: Następujące opcje sterują tym, co dzieje się z nietrwałymi komunikatami, które zostały wysłane do klienta przed zgłoszeniem żądania przez aplikację i nie zostały jeszcze wykorzystane przez aplikację. Te komunikaty są przechowywane w buforze odczytu z wyprzedzeniem klienta, który oczekuje na żądanie aplikacji i może zostać usunięty lub skonsumowany z kolejki przed zakończeniem operacji MQCLOSE.

MQCO_IMMEDIATE
Obiekt jest zamykany natychmiast, a wszystkie komunikaty, które zostały wysłane do klienta przed zażądaniem ich przez aplikację, są usuwane i nie są dostępne do wykorzystania przez żadną aplikację. Jest to wartość domyślna.

MQCO_QUIESCE,
Żądanie zamknięcia obiektu jest wykonywane, ale jeśli jakiekolwiek komunikaty, które zostały wysłane do klienta przed zażądaniem ich przez aplikację, nadal znajdują się w buforze odczytu z wyprzedzeniem klienta, wywołanie MQCLOSE zwraca ostrzeżenie MQRC_READ_AHEAD_MSGS i uchwyt obiektu pozostaje poprawny.

Aplikacja może kontynuować używanie uchwytu obiektu do pobierania komunikatów do czasu, aż nie będzie już więcej dostępnych, a następnie ponownie zamknąć obiekt. Nie są wysyłane żadne komunikaty do klienta przed aplikacją, która je zażądała. Odczyt z wyprzedzeniem jest teraz wyłączony.

Zaleca się, aby aplikacje używały komendy MQCO_QUIESCE zamiast próby osiągnięcia punktu, w którym nie ma więcej komunikatów w buforze odczytu z wyprzedzeniem klienta, ponieważ komunikat może nadejść między ostatnim wywołaniem MQGET a następnym wywołaniem MQCLOSE, który zostałby odrzucony, gdyby użyto komendy MQCO_IMMEDIATE.

Jeśli komenda MQCLOSE z opcją MQCO_QUIESCE jest wykonywana z poziomu asynchronicznej funkcji zwrotnej, stosowane jest takie samo zachowanie odczytu komunikatów z wyprzedzeniem. Jeśli zostanie zwrócone ostrzeżenie MQRC_READ_AHEAD_MSGS, funkcja zwrotna zostanie wywołana co najmniej jeden raz. Gdy ostatni pozostały komunikat, który został odczytany z wyprzedzeniem, został przekazany do funkcji zwrotnej, pole ConsumerFlags MQCBC jest ustawione na wartość MQCBCF_READA_BUFFER_EMPTY.

Opcja domyślna: Jeśli nie jest wymagana żadna z opisanych wcześniej opcji, można użyć następującej opcji:

MQCO_BRAK
Nie jest wymagane opcjonalne przetwarzanie zamknięcia.

Należy to określić dla:

- Obiekty inne niż kolejki
- Kolejki predefiniowane
- Tymczasowe kolejki dynamiczne (ale tylko w tych przypadkach, w których *Hobj* nie jest uchwytem zwróconym przez wywołanie MQOPEN, które utworzyło kolejkę).
- Lista dystrybucyjna

We wszystkich powyższych przypadkach obiekt jest zachowywany i nie jest usuwany.

Jeśli ta opcja jest określona dla tymczasowej kolejki dynamicznej:

- Kolejka zostanie usunięta, jeśli została utworzona przez wywołanie MQOPEN, które zwróciło kod *Hobj* . wszystkie komunikaty znajdujące się w kolejce są czyszczone.
- We wszystkich innych przypadkach kolejka (i wszystkie znajdujące się na niej komunikaty) są zachowywane.

Jeśli ta opcja jest określona dla trwałej kolejki dynamicznej, kolejka jest zachowywana i nie jest usuwana.

W systemie z/OS, jeśli kolejka jest kolejką dynamiczną, która została logicznie usunięta i jest to jej ostatni uchwyt, kolejka jest fizycznie usuwana. Więcej informacji na ten temat zawiera sekcja “Użycie notatek” na stronie 675.

CompCode
Typ: MQLONG-output

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-output

Wymienione kody przyczyny są tymi, które menedżer kolejek może zwrócić dla parametru **Reason** .

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

GRUPA MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

MQRC_READ_AHEAD_MSGS
(nnnn, X'xxx ') Klient odczytał komunikaty z wyprzedzeniem, które nie zostały jeszcze wykorzystane przez aplikację.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_NIEDOSTĘPNE
(2345, X' 929 ') Obiekt sprzęgania nie jest dostępny.

MQRC_CF_STRUC_FAILED
(2373, X' 945 ') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') W wywołaniu MQOPEN lub MQCLOSE opcja nie jest poprawna dla typu obiektu.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje niepoprawne lub niespójne.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_NOT_EMPTY
(2055, X'807 ') Kolejka zawiera jeden lub więcej komunikatów lub niezatwierdzone żądania umieszczania lub pobierania.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_SECURITY_ERROR
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Gdy aplikacja wysyła wywołanie MQDISC lub kończy działanie w sposób normalny lub nienormalny, wszystkie obiekty, które zostały otwarte przez aplikację i są nadal otwarte, są zamykane automatycznie z opcją MQCO_NONE.
2. Jeśli zamykany obiekt jest *kolejką*, mają zastosowanie następujące punkty:
   - Jeśli operacje w kolejce są wykonywane w ramach jednostki pracy, można zamknąć kolejkę przed lub po wystąpieniu punktu synchronizacji bez wpływu na wynik punktu synchronizacji. Jeśli kolejka jest wyzwalana, wycofanie zmian przed zamknięciem kolejki może spowodować wysłanie komunikatu wyzwalacza. Więcej informacji na temat komunikatów wyzwalacza zawiera sekcja Właściwości komunikatów wyzwalacza.
   - Jeśli kolejka została otwarta z opcją MQOO_BROWSE, kursor przeglądania jest niszczony. Jeśli kolejka zostanie ponownie otwarta z opcją MQOO_BROWSE, zostanie utworzony nowy kursor przeglądania (patrz MQOO_BROWSE).
   - Jeśli komunikat jest aktualnie zablokowany dla tego uchwytu w czasie wywołania MQCLOSE, blokada jest zwalniana (patrz MQGMO_LOCK).
   - W systemie z/OS, jeśli dla zamykanego uchwytu kolejki istnieje żądanie MQGET z opcją MQGMO_SET_SIGNAL, żądanie jest anulowane (patrz sekcja MQGMO_SET_SIGNAL). Nie ma to wpływu na żądania sygnału dla tej samej kolejki, ale złożone dla różnych uchwytów (*Hobj*), chyba że kolejka dynamiczna jest usuwana (w tym przypadku są one również anulowane).
3. Poniższe punkty mają zastosowanie, jeśli zamykany obiekt jest *kolejką dynamiczną* (trwałą lub tymczasową):
   - W przypadku kolejki dynamicznej można określić opcje MQCO_DELETE i MQCO_DELETE_PURGE niezależnie od opcji określonych w odpowiednim wywołaniu MQOPEN.
   - Po usunięciu kolejki dynamicznej wszystkie wywołania MQGET z opcją MQGMO_WAIT, które są zaległe dla kolejki, są anulowane i zwracany jest kod przyczyny MQRC_Q_DELETED. Patrz MQGMO_WAIT.

     Chociaż aplikacje nie mogą uzyskać dostępu do usuniętej kolejki, kolejka nie jest usuwana z systemu, a powiązane z nią zasoby nie są zwalniane, dopóki wszystkie uchwyty odwołujące się do kolejki nie

zostaną zamknięte, a wszystkie jednostki pracy wpływające na kolejkę nie zostaną zatwierdzone lub wycofane.

W systemie z/OSkolejka, która została logicznie usunięta, ale nie została jeszcze usunięta z systemu, uniemożliwia utworzenie nowej kolejki o takiej samej nazwie, jak usunięta kolejka. W tym przypadku wywołanie MQOPEN kończy się niepowodzeniem z kodem przyczyny MQRC_NAME_IN_USE. Taka kolejka może być nadal wyświetlana przy użyciu komend MQSC, nawet jeśli aplikacje nie mają do niej dostępu.

- Po usunięciu trwałej kolejki dynamicznej, jeśli uchwyt *Hobj* określony w wywołaniu MQCLOSE nie jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, wykonywane jest sprawdzenie, czy identyfikator użytkownika użyty do sprawdzenia poprawności wywołania MQOPEN ma uprawnienia do usunięcia kolejki. Jeśli w wywołaniu MQOPEN podano opcję MQOO_ALTERNATE_USER_AUTHORITY, sprawdzany identyfikator użytkownika to *AlternateUserId*.

  To sprawdzenie nie jest wykonywane, jeśli:

  – Podany uchwyt jest zwracany przez wywołanie MQOPEN, które utworzyło kolejkę.

  – Usuwana kolejka jest tymczasową kolejką dynamiczną.

- Po zamknięciu tymczasowej kolejki dynamicznej, jeśli uchwyt *Hobj* określony w wywołaniu MQCLOSE jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, zostanie ona usunięta. Dzieje się tak niezależnie od opcji zamykania określonych w wywołaniu MQCLOSE. Jeśli w kolejce znajdują się komunikaty, są one usuwane i nie są generowane żadne komunikaty raportów.

  Jeśli istnieją niezatwierdzone jednostki pracy, które mają wpływ na kolejkę, kolejka i jej komunikaty są nadal usuwane, ale jednostki pracy nie kończą się niepowodzeniem. Jednak, jak opisano wcześniej, zasoby powiązane z jednostkami pracy nie są zwalniane, dopóki żadna z jednostek pracy nie zostanie zatwierdzona lub wycofana.

4. Poniższe punkty mają zastosowanie, jeśli zamykany obiekt jest *listą dystrybucyjną*:

   - Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest MQCO_NONE; wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OPTIONS_ERROR lub MQRC_OPTION_NOT_VALID_FOR_TYPE, jeśli określono inne opcje.
   - Po zamknięciu listy dystrybucyjnej indywidualne kody zakończenia i kody przyczyny nie są zwracane dla kolejek znajdujących się na liście. Do celów diagnostycznych dostępne są tylko parametry **CompCode** i **Reason** wywołania.

     Jeśli wystąpi błąd podczas zamykania jednej z kolejek, menedżer kolejek kontynuuje przetwarzanie i próbuje zamknąć pozostałe kolejki na liście dystrybucyjnej. Parametry **CompCode** i **Reason** wywołania są ustawione w celu zwrócenia informacji opisujących niepowodzenie. Kod zakończenia może mieć wartość MQCC_FAILED, nawet jeśli większość kolejek została pomyślnie zamknięta. Kolejka, w której wystąpił błąd, nie została zidentyfikowana.

     Jeśli awaria wystąpi w więcej niż jednej kolejce, nie jest zdefiniowana, która awaria jest zgłaszana w parametrach **CompCode** i **Reason** .

Wywołanie C

```
MQCLOSE (Hconn, &Hobj, Options, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
     CALL 'MQCLOSE' USING HCONN, HOBJ, OPTIONS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQCLOSE (Hconn, Hobj, Options, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQCLOSE,(HCONN,HOBJ,OPTIONS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

```
MQCLOSE Hconn, Hobj, Options, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-zatwierdzanie zmian

Wywołanie MQCMIT wskazuje menedżerowi kolejek, że aplikacja osiągnęła punkt synchronizacji oraz że wszystkie operacje pobierania i umieszczania komunikatów, które wystąpiły od ostatniego punktu synchronizacji, mają zostać utrwalone.

Komunikaty umieszczone jako część jednostki pracy są udostępniane innym aplikacjom; komunikaty pobrane jako część jednostki pracy są usuwane.

- z/OS W systemie z/OSwywołanie jest używane tylko przez programy wsadowe (w tym programy DL/I w języku IMS).

Składnia

MQCMIT (*Hconn*, *CompCode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Wymienione kody przyczyny są tymi, które menedżer kolejek może zwrócić dla parametru **Reason** .

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_BACKED_OUT
(2003, X'7D3') Jednostka pracy wycofała się.

MQRC_OUTCOME_PENDING (oczekiwanie na wynik MQ)
(2124, X'84C') Wynik operacji zatwierdzania jest w toku.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CALL_PRZERWANE
(2549, X'9F5') MQPUT lub MQCMIT zostało przerwane i przetwarzanie ponownego połączenia nie może ponownie ustanowić określonego wyniku.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

MQRC_OUTCOME_MIXED
(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Po ponownym nawiązaniu połączenia wystąpił błąd podczas przywracania uchwytów dla połączenia z możliwością ponownego połączenia.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Tego wywołania należy używać tylko wtedy, gdy sam menedżer kolejek koordynuje jednostkę pracy. Może to być:
   - Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
   - Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby IBM MQ .

   Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja “MQBEGIN-Rozpoczęcie jednostki pracy” na stronie 651.
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiedniego wywołania zatwierdzenia zamiast MQCMIT. Środowisko może również obsługiwać niejawne zatwierdzanie spowodowane przez normalne zakończenie działania aplikacji.
   - W systemie z/OSnależy użyć następujących wywołań:
     – Programy wsadowe (w tym programy IMS w języku DL/I) mogą używać wywołania MQCMIT, jeśli jednostka pracy ma wpływ tylko na zasoby IBM MQ . Jeśli jednak jednostka pracy ma wpływ zarówno na zasoby IBM MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRCMIT udostępnianego przez usługę RRS (z/OS

Recoverable Resource Service). Wywołanie SRRCMIT zatwierdza zmiany w zasobach należących do menedżerów zasobów, dla których włączono koordynację RRS.

- Aplikacje CICS muszą używać komendy EXEC CICS SYNCPOINT do jawnego zatwierdzania jednostki pracy. Alternatywnie zakończenie transakcji powoduje niejawne zatwierdzenie jednostki pracy. Wywołania MQCMIT nie można używać dla aplikacji CICS .
- Aplikacje IMS (inne niż programy wsadowe DL/I) muszą używać wywołań IMS , takich jak GU i CHKP , do zatwierdzenia jednostki pracy. Wywołania MQCMIT nie można używać dla aplikacji IMS (innych niż programy wsadowe DL/I).

- W systemie IBM itego wywołania należy używać dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie została wydana dla zadania.

3. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawiera sekcja Uwagi dotyczące użycia usługi MQDISC .
4. Gdy aplikacja umieszcza lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Te informacje są powiązane z uchwytem kolejki i obejmują takie elementy, jak:

   - Wartości pól *GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags* w strukturze MQMD.
   - Określa, czy komunikat jest częścią jednostki pracy.
   - Dla wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

   Po zatwierdzeniu jednostki pracy menedżer kolejek zachowuje informacje o grupie i segmencie, a aplikacja może kontynuować umieszczanie lub pobieranie komunikatów w bieżącej grupie komunikatów lub w komunikacie logicznym.

   Zachowanie informacji o grupie i segmencie po zatwierdzeniu jednostki pracy umożliwia aplikacji rozmieszczenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy. Użycie kilku jednostek pracy jest korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną pamięć kolejki. Jednak w przypadku wystąpienia awarii systemu aplikacja musi zachować wystarczającą ilość informacji, aby zrestartować umieszczanie lub pobieranie komunikatów w odpowiednim miejscu. Szczegółowe informacje na temat sposobu restartowania w odpowiednim punkcie po awarii systemu można znaleźć w sekcji MQPMO_LOGICAL_ORDER i w sekcji MQGMO_LOGICAL_ORDER.

   Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

5. Jednostka pracy ma taki sam zasięg jak uchwyt połączenia; wszystkie wywołania IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wykonane przy użyciu innego uchwytu połączenia (na przykład wywołania wykonane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączeń zawiera opis parametru **Hconn** w sekcji MQCONN.
6. To wywołanie ma wpływ tylko na komunikaty, które zostały umieszczone lub pobrane jako część bieżącej jednostki pracy.
7. Aplikacja długotrwała, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wywołania wycofanego, może zapełniać kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zapobiec takiej sytuacji, administrator musi ustawić atrybut menedżera kolejek **MaxUncommittedMsgs** na wartość, która jest na tyle niska, aby zapobiec zapełnianiu kolejek przez aplikacje działające w trybie niepewnym, ale na tyle wysoka, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.
8. ALW W systemach AIX, Linux, and Windows , jeśli parametr **Reason** ma wartość MQRC_CONNECTION_BROKEN (z parametrem *CompCode* o wartości MQCC_FAILED) lub MQRC_UNEXPECTED_ERROR, możliwe, że jednostka pracy została pomyślnie zatwierdzona.

Wywołanie C

```
MQCMIT (Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Wywołanie COBOL

```
CALL 'MQCMIT' USING HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQCMIT (Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Wywołanie programu High Level Assembler

```
CALL MQCMIT,(HCONN,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Wywołanie Visual Basic

```
MQCMIT Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQCONN-Połącz z menedżerem kolejek

Wywołanie MQCONN łączy aplikację z menedżerem kolejek.

Udostępnia on uchwyt połączenia menedżera kolejek, który jest używany przez aplikację w kolejnych wywołaniach kolejkowania komunikatów.

- W systemie z/OSaplikacje CICS nie muszą wywoływać tego wywołania. Te aplikacje są automatycznie połączone z menedżerem kolejek, z którym jest połączony system CICS . Jednak wywołania MQCONN i MQDISC są nadal akceptowane przez aplikacje CICS .
- W systemie IBM iaplikacje muszą używać wywołania MQCONN lub MQCONNX w celu nawiązania połączenia z menedżerem kolejek, a wywołania MQDISC w celu rozłączenia się z menedżerem kolejek.

Połączenie klienta nie może być nawiązywane tylko na serwerze, a połączenie lokalne nie może być nawiązywane tylko na kliencie.

Składnia

MQCONN (*QMgrName*, *Hconn*, *CompCode*, *Przyczyna*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Jest to nazwa menedżera kolejek, z którym aplikacja chce się połączyć. Nazwa może zawierać następujące znaki:

- Wielkie litery (od A do Z)
- Małe litery (od a do z)
- Cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani wewnętrznych odstępów, ale może zawierać końcowe odstępy. Znak o kodzie zero może być używany do wskazania końca istotnych danych w nazwie; znak o kodzie zero i wszystkie następujące po nim znaki są traktowane jako odstępy. W podanych środowiskach obowiązują następujące ograniczenia:

- W systemach używających kodu EBCDIC Katakana nie można używać małych liter.
- W systemie z/OSnazwy rozpoczynające się lub kończące się znakiem podkreślenia nie mogą być przetwarzane przez operacje i panele sterowania. Z tego powodu należy unikać takich nazw.
- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent należy ująć w cudzysłów, jeśli zostały podane w komendach. Nie należy podawać tych cudzysłowów w parametrze **QMgrName** .

Jeśli nazwa składa się wyłącznie z odstępów, używana jest nazwa *domyślnego* menedżera kolejek. Należy jednak pamiętać o użyciu pustych nazw menedżerów kolejek opisanych w sekcji dotyczącej aplikacji IBM MQ MQI client .

Nazwa określona dla parametru *QMgrName* musi być nazwą *możliwego do połączenia* menedżera kolejek lub, jeśli używane są grupy menedżerów kolejek, nazwą grupy menedżerów kolejek.

W systemie z/OSmenedżery kolejek, z którymi można nawiązać połączenie, są określane przez środowisko:

- W przypadku systemu CICSmożna używać tylko menedżera kolejek, z którym połączony jest system CICS . Parametr **QMgrName** musi być nadal określony, ale jego wartość jest ignorowana; odpowiednie są znaki odstępu.
- W przypadku systemu IMSmożna nawiązać połączenie tylko z menedżerami kolejek wymienionymi w tabeli definicji podsystemu (CSQQDEFV), *i* wymienionymi w tabeli SSM w pliku IMS(patrz uwaga dotycząca użycia 6).

- W przypadku zadań wsadowych systemu z/OS i TSO tylko menedżery kolejek rezydujące w tym samym systemie co aplikacja są dostępne do połączenia (patrz uwaga dotycząca użycia 6).

Grupy współużytkowania kolejek:W systemach, w których istnieje kilka menedżerów kolejek skonfigurowanych w celu utworzenia grupy współużytkowania kolejek, nazwę grupy współużytkowania kolejek można określić dla parametru *QMgrName* zamiast nazwy menedżera kolejek. Umożliwia to aplikacji nawiązanie połączenia z *dowolnym* menedżerem kolejek, który jest dostępny w grupie współużytkowania kolejek i znajduje się na tym samym obrazie systemu z/OS co aplikacja. System można również skonfigurować w taki sposób, aby program *QMgrName* nawiązywał połączenie z grupą współużytkowania kolejek zamiast z domyślnym menedżerem kolejek.

Jeśli parametr *QMgrName* określa nazwę grupy współużytkowania kolejek, ale w systemie istnieje również menedżer kolejek o tej nazwie, nawiązywane jest połączenie z tym drugim menedżerem kolejek, a nie z tym pierwszym. Tylko wtedy, gdy to połączenie nie powiedzie się, nawiązywane jest połączenie z jednym z menedżerów kolejek w grupie współużytkowania kolejek.

Jeśli połączenie zostanie pomyślnie nawiązane, można użyć uchwytu zwróconego przez wywołanie MQCONN lub MQCONNX, aby uzyskać dostęp do *wszystkich* zasobów (zarówno współużytkowanych, jak i niewspółużytkowanych) należących do menedżera kolejek, z którym nawiązano połączenie. Dostęp do tych zasobów podlega typowym kontrolom autoryzacji.

Jeśli aplikacja wysyła dwa wywołania MQCONN lub MQCONNX w celu nawiązania połączeń współbieżnych, a jedno lub oba wywołania określają nazwę grupy współużytkowania kolejek, drugie wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_ALREADY_CONNECTED, gdy nawiązuje połączenie z tym samym menedżerem kolejek co pierwsze wywołanie.

Grupy współużytkowania kolejek są obsługiwane tylko w systemie z/OS. Połączenie z grupą współużytkowania kolejek jest obsługiwane tylko w środowisku wsadowym, RRS, CICSi TSO.
W przypadku systemu CICSmożna używać tylko grupy współużytkowania kolejek, z którą połączony jest system CICS . Nadal należy podać parametr **QMgrName** , ale jego wartość jest ignorowana; odpowiednie są znaki odstępu.

Ostrzeżenie: Program IMS nie może połączyć się z grupą współużytkowania kolejek.

Aplikacje systemu IBM MQ MQI client: w przypadku aplikacji systemu IBM MQ MQI client podejmowana jest próba nawiązania połączenia dla każdej definicji kanału połączenia klienckiego o określonej nazwie menedżera kolejek do momentu pomyślnego nawiązania połączenia. Jednak menedżer kolejek musi mieć taką samą nazwę, jak określona nazwa. Jeśli zostanie podana pusta nazwa, podejmowana jest próba nawiązania połączenia z każdym kanałem połączenia klienckiego z pustą nazwą menedżera kolejek do momentu pomyślnego zakończenia. W takim przypadku nie jest sprawdzana rzeczywista nazwa menedżera kolejek.

Aplikacje klienckie IBM MQ nie są obsługiwane w produkcie z/OS, ale produkt z/OS może działać jako serwer IBM MQ , z którym mogą łączyć się aplikacje klienckie IBM MQ .

IBM MQ MQI client Grupy menedżerów kolejek: Jeśli podana nazwa rozpoczyna się od gwiazdki (*), menedżer kolejek, z którym nawiązywane jest połączenie, może mieć inną nazwę niż nazwa określona przez aplikację. Podana nazwa (bez gwiazdki) definiuje *grupę* menedżerów kolejek zakwalifikowanych do połączenia. Implementacja wybiera jedną z grupy, próbując kolejno każdą z nich, aż do znalezienia jednej, z którą można nawiązać połączenie. Kolejność prób nawiązania połączenia zależy od wagi kanału klienta i wartości powinowactwa połączenia kanałów kandydujących. Jeśli żaden z menedżerów kolejek w grupie nie jest dostępny dla połączenia, wywołanie nie powiedzie się. Każdy menedżer kolejek jest wypróbowany tylko raz. Jeśli dla nazwy podano samą gwiazdkę, zostanie użyta domyślna grupa menedżerów kolejek zdefiniowana przez implementację.

Grupy menedżerów kolejek są obsługiwane tylko w przypadku aplikacji działających w środowisku klienta produktu MQ. Wywołanie nie powiedzie się, jeśli aplikacja niekliencka określi nazwę menedżera kolejek rozpoczynającą się od gwiazdki. Grupa jest definiowana przez udostępnienie kilku definicji kanału połączenia klienckiego o tej samej nazwie menedżera kolejek (określonej nazwie bez gwiazdki) w celu komunikowania się z każdym menedżerem kolejek w grupie. Grupa domyślna jest definiowana przez podanie co najmniej jednej definicji kanału połączenia klienckiego, z której

każda ma pustą nazwę menedżera kolejek (podanie nazwy all-blank ma taki sam skutek, jak podanie pojedynczej gwiazdki dla nazwy aplikacji klienckiej).

Po nawiązaniu połączenia z jednym menedżerem kolejek grupy aplikacja może określić odstępy w typowy sposób w polach nazwy menedżera kolejek w komunikacie i deskryptorach obiektów, co oznacza nazwę menedżera kolejek, z którym połączona jest aplikacja (*menedżer kolejek lokalnych*). Jeśli aplikacja musi znać tę nazwę, należy użyć wywołania MQINQ w celu uzyskania informacji o atrybucie menedżera kolejek **QMgrName** .

Umieszczenie znaku gwiazdki na początku nazwy połączenia oznacza, że aplikacja nie jest zależna od połączenia z określonym menedżerem kolejek w grupie. Odpowiednie zastosowania to:

- Aplikacje, które umieszczają komunikaty, ale ich nie dostają.
- Aplikacje, które umieszczają komunikaty żądań, a następnie pobierają komunikaty odpowiedzi z *tymczasowej kolejki dynamicznej* .

Nieodpowiednie aplikacje to te, które muszą pobrać komunikaty z określonej kolejki w danym menedżerze kolejek. Takie aplikacje nie mogą poprzedzać nazwy gwiazdką.

Jeśli zostanie podana gwiazdka, maksymalna długość pozostałej części nazwy wynosi 47 znaków.

Długość tego parametru jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

hconn

Typ: MQHCONN-output

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Należy ją określić we wszystkich kolejnych wywołaniach kolejkowania komunikatów wysyłanych przez aplikację. Przestaje być ona poprawna po wywołaniu wywołania MQDISC lub po zakończeniu jednostki przetwarzania definiującej zasięg uchwytu.

Produkt IBM MQ dostarcza teraz bibliotekę mqm z pakietami klienckimi, a także pakietami serwera. Oznacza to, że gdy wykonywane jest wywołanie MQI znalezione w bibliotece mqm, sprawdzany jest typ połączenia w celu sprawdzenia, czy jest to połączenie klienta lub serwera, a następnie wykonywane jest poprawne wywołanie bazowe. Dlatego wyjście, do którego przekazano *Hconn* , może być teraz powiązane z biblioteką mqm, ale używane w instalacji klienta.

Zasięg uchwytu: Zasięg zwróconego uchwytu zależy od wywołania używanego do nawiązania połączenia z menedżerem kolejek (MQCONN lub MQCONNX). Jeśli używane jest wywołanie MQCONNX, zasięg uchwytu zależy również od opcji MQCNO_HANDLE_SHARE_ * określonej w polu *Options* struktury MQCNO.

- Jeśli wywołanie to MQCONN lub podano opcję MQCNO_HANDLE_SHARE_NONE, zwrócony uchwyt jest uchwytem *niewspółużytkowanym* .

  Zasięg niewspółużytkowanego uchwytu to najmniejsza jednostka przetwarzania równoległego obsługiwana przez platformę, na której działa aplikacja (szczegółowe informacje zawiera sekcja Tabela 546 na stronie 685). Uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołano wywołanie.
- Jeśli zostanie podana opcja MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, zwracany uchwyt jest uchwytem *współużytkowanym* .

  Zasięg uchwytu współużytkowanego to proces, który jest właścicielem wątku, z którego wywołano wywołanie. Uchwyt może być używany z dowolnego wątku, który należy do tego procesu. Nie wszystkie platformy obsługują wątki.
- Jeśli wywołanie MQCONN lub MQCONNX nie powiedzie się z kodem zakończenia równym MQCC_FAILED, wartość Hconn jest niezdefiniowana.

Tabela 546. Zasięg niewspółużytkowanych uchwytów na różnych platformach	
Platforma	**Zasięg niewspółużytkowanego uchwytu**
z/OS	• CICS: zadanie CICS • IMS: zadanie aż do następnego punktu synchronizacji (z wyłączeniem podzadań zadania) • Zadanie wsadowe z/OS i TSO: zadanie (z wyłączeniem podzadań zadania)
IBM i	Praca
AIX and Linux	Wątek
32-bitowe aplikacje Windows	Wątek
64-bitowe aplikacje Windows	Wątek

W systemie z/OS dla aplikacji CICS zwracana jest wartość:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

POŁĄCZONO MQRC_ALREADY_CONNECTED
(2002, X'7D2') Aplikacja jest już podłączona.

BŁĄD ŁADOWANIA MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') Nie można załadować wyjścia obciążenia klastra.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') Protokół SSL został już zainicjowany.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851 ') Nie można załadować modułu połączenia adaptera.

BŁĄD MQRC_ADAPTER_DEFS_ERROR
(2131, X'853 ') Moduł definicji podsystemu adaptera jest niepoprawny.

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854 ') Nie można załadować modułu definicji podsystemu adaptera.

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ADAPTER_STORAGE_NIEDOBÓR
(2127, X'84F') Niewystarczająca ilość pamięci dla adaptera.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Inny menedżer kolejek jest już połączony.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

MQRC_API_EXIT_INIT_BŁĄD
(2375, X' 947 ') Inicjowanie wyjścia funkcji API nie powiodło się.

BŁĄD MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') Zakończenie wyjścia funkcji API nie powiodło się.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ID_KOND_MQRC_W_UŻYCIU
(2160, X'870 ') Identyfikator połączenia jest już używany.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD POŁĄCZENIA MQRC_CONNECTION_
(2273, X'8E1') Błąd podczas przetwarzania wywołania MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Występuje w wywołaniu MQCONN lub MQCONNX, gdy menedżer kolejek nie może udostępnić połączenia żądanego typu połączenia w bieżącej instalacji. Nie można nawiązać połączenia z klientem tylko na serwerze. Połączenie lokalne nie może być nawiązywane tylko na kliencie.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Błąd konfiguracji sprzętu szyfrującego.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') Istnieje koordynator odtwarzania.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

Dodatkowo w wywołaniu MQCONNX, przekazując blok kontrolny “MQCSP-parametry zabezpieczeń” na stronie 341 z aplikacji CICS lub IMS .

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_HOST_NOT_AVAILABLE (niedostępny)
(2538, X'9EA') Klient wysłał wywołanie MQCONN w celu nawiązania połączenia z menedżerem kolejek, ale próba przydzielenia konwersacji do systemu zdalnego nie powiodła się.

MQRC_INSTALLATION_MISMATCH (Niezgodność instalacji MQ)
(2583, X'A17') Niezgodność między instalacją menedżera kolejek i wybraną biblioteką.

BŁĄD MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') Repozytorium kluczy jest niepoprawne.

MQRC_MAX_CONNS_LIMIT_REACHED
(2025, X'7E9') Osiągnięto maksymalną liczbę połączeń.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

MQRC_OPEN_FAILED,
(2137, X'859 ') Obiekt nie został pomyślnie otwarty.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_SECURITY_ERROR
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

MQRC_SSL_INITIALIZATION_ERROR (błąd)
(2393, X' 959 ') Błąd inicjowania SSL.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Menedżer kolejek, z którym nawiązywane jest połączenie przy użyciu wywołania MQCONN, jest nazywany *lokalnym menedżerem kolejek*.
2. Kolejki, których właścicielem jest menedżer kolejek lokalnych, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie z nich komunikatów.

   Kolejki współużytkowane, których właścicielem jest grupa współużytkowania kolejek, do której należy menedżer kolejek lokalnych, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie z nich komunikatów.

   Kolejki należące do menedżerów kolejek zdalnych są wyświetlane jako kolejki zdalne. Możliwe jest umieszczanie komunikatów w tych kolejkach, ale nie pobieranie komunikatów z tych kolejek.
3. Jeśli działanie menedżera kolejek zakończy się niepowodzeniem podczas działania aplikacji, aplikacja musi ponownie wywołać komendę MQCONN, aby uzyskać nowy uchwyt połączenia do użycia w kolejnych wywołaniach programu IBM MQ . Aplikacja może okresowo wywoływać wywołanie MQCONN, dopóki wywołanie nie zakończy się powodzeniem.

   Jeśli aplikacja nie jest pewna, czy jest połączona z menedżerem kolejek, może bezpiecznie wywołać wywołanie MQCONN w celu uzyskania uchwytu połączenia. Jeśli aplikacja jest już połączona, zwrócony uchwyt jest taki sam, jak zwrócony przez poprzednie wywołanie MQCONN, ale z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_ALREADY_CONNECTED.
4. Po zakończeniu używania przez aplikację wywołań IBM MQ aplikacja musi użyć wywołania MQDISC w celu rozłączenia się z menedżerem kolejek.

5. Jeśli wywołanie MQCONN nie powiedzie się z kodem zakończenia równym MQCC_FAILED, wartość Hconn jest niezdefiniowana.
6. W systemie z/OS:
   - Aplikacje wsadowe, TSO i IMS muszą wywołać MQCONN, aby użyć innych wywołań IBM MQ . Te aplikacje mogą łączyć się współbieżnie z więcej niż jednym menedżerem kolejek.

     Jeśli działanie menedżera kolejek nie powiedzie się, aplikacja musi ponownie wywołać wywołanie po zrestartowaniu menedżera kolejek w celu uzyskania nowego uchwytu połączenia.

     Mimo że aplikacje IMS mogą wywoływać wywołania MQCONN wielokrotnie, nawet jeśli są już połączone, nie jest to zalecane w przypadku programów przetwarzania komunikatów (MPP) w trybie z połączeniem.
   - Aplikacje CICS nie muszą wywoływać wywołania MQCONN w celu użycia innych wywołań IBM MQ , ale mogą to zrobić w razie potrzeby. Akceptowane jest zarówno wywołanie MQCONN, jak i wywołanie MQDISC. Nie jest jednak możliwe współbieżne nawiązanie połączenia z więcej niż jednym menedżerem kolejek.

     Jeśli działanie menedżera kolejek nie powiedzie się, aplikacje te zostaną automatycznie ponownie połączone podczas restartowania menedżera kolejek, dlatego nie trzeba wywoływać wywołania MQCONN.
7. W systemie z/OS, aby zdefiniować dostępne menedżery kolejek:
   - W przypadku aplikacji wsadowych programiści systemu mogą użyć makra CSQBDEF, aby utworzyć moduł (CSQBDEFV) definiujący domyślną nazwę menedżera kolejek lub nazwę grupy współużytkowania kolejek.
   - W przypadku aplikacji IMS programiści systemu mogą użyć makra CSQQDEFX do utworzenia modułu (CSQQDEFV), który definiuje nazwy dostępnych menedżerów kolejek i określa domyślny menedżer kolejek.

     Ponadto każdy menedżer kolejek musi być zdefiniowany w regionie sterującym IMS oraz w każdym regionie zależnym uzyskującego dostęp do tego menedżera kolejek. W tym celu należy utworzyć element podsystemu w IMS.Biblioteka PROCLIB i zidentyfikuj element podsystemu w odpowiednich regionach IMS . Jeśli aplikacja podejmie próbę nawiązania połączenia z menedżerem kolejek, który nie jest zdefiniowany w elemencie podsystemu dla swojego regionu IMS , aplikacja zostanie zakończona awaryjnie.

   z/OS Więcej informacji na temat używania tych makr zawiera sekcja Makra przeznaczone do użytku przez klienta.
8. W systemie IBM iprogramy zakończone nieprawidłowo nie są automatycznie odłączane od menedżera kolejek. Napisz aplikacje, aby umożliwić wywołanie MQCONN lub MQCONNX zwracające kod zakończenia MQCC_WARNING i kod przyczyny MQRC_ALREADY_CONNECTED. W takiej sytuacji należy użyć uchwytu połączenia zwróconego w normalnej sytuacji.

Wywołanie C

```
MQCONN (QMgrName, &Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQCHAR48  QMgrName;  /* Name of queue manager */
MQHCONN   Hconn;     /* Connection handle */
MQLONG    CompCode;  /* Completion code */
MQLONG    Reason;    /* Reason code qualifying CompCode */
```

Wywołanie COBOL

```
CALL 'MQCONN' USING QMGRNAME, HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQCONN (QMgrName, Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQCONN,(QMGRNAME,HCONN,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
QMGRNAME  DS  CL48  Name of queue manager
HCONN     DS  F     Connection handle
COMPCODE  DS  F     Completion code
REASON    DS  F     Reason code qualifying COMPCODE
```

Wywołanie Visual Basic

```
MQCONN QMgrName, Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-Połącz z menedżerem kolejek (rozszerzone)

Wywołanie MQCONNX łączy aplikację z menedżerem kolejek. Udostępnia on uchwyt połączenia menedżera kolejek, który jest używany przez aplikację w kolejnych wywołaniach programu IBM MQ .

Wywołanie MQCONNX jest podobne do wywołania MQCONN, z tą różnicą, że MQCONNX umożliwia określenie opcji sterujących sposobem działania wywołania.

- To wywołanie jest obsługiwane we wszystkich systemach IBM MQ i klientach IBM MQ podłączonych do tych systemów.

Połączenie klienta nie może być nawiązywane tylko na serwerze, a połączenie lokalne nie może być nawiązywane tylko na kliencie.

Składnia

MQCONNX (*QMgrName*, *ConnectOpts*, *Hconn*, *CompCode*, *Przyczyna*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Szczegółowe informacje zawiera opis parametru **QMgrName** w sekcji "MQCONN-Połącz z menedżerem kolejek" na stronie 682 .

ConnectOpts
Typ: MQCNO-input/output

Szczegółowe informacje można znaleźć w sekcji "MQCNO-opcje połączenia" na stronie 321.

hconn
Typ: MQHCONN-output

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Należy ją określić we wszystkich kolejnych wywołaniach kolejkowania komunikatów wysyłanych przez aplikację. Przestaje być ona poprawna po wywołaniu wywołania MQDISC lub po zakończeniu jednostki przetwarzania definiującej zasięg uchwytu.

Produkt IBM MQ dostarcza teraz bibliotekę mqm z pakietami klienckimi, a także pakietami serwera. Oznacza to, że gdy wykonywane jest wywołanie MQI znalezione w bibliotece mqm, sprawdzany jest typ połączenia w celu sprawdzenia, czy jest to połączenie klienta lub serwera, a następnie wykonywane jest poprawne wywołanie bazowe. Dlatego wyjście, do którego przekazano *Hconn* , może być teraz powiązane z biblioteką mqm, ale używane w instalacji klienta.

Zasięg uchwytu: Zasięg zwróconego uchwytu zależy od wywołania używanego do nawiązania połączenia z menedżerem kolejek (MQCONN lub MQCONNX). Jeśli używane jest wywołanie MQCONNX, zasięg uchwytu zależy również od opcji MQCNO_HANDLE_SHARE_ * określonej w polu *Options* struktury MQCNO.

- Jeśli wywołanie to MQCONN lub podano opcję MQCNO_HANDLE_SHARE_NONE, zwrócony uchwyt jest uchwytem *niewspółużytkowanym* .

  Zasięg niewspółużytkowanego uchwytu to najmniejsza jednostka przetwarzania równoległego obsługiwana przez platformę, na której działa aplikacja (szczegółowe informacje zawiera sekcja Tabela 547 na stronie 691). Uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołano wywołanie.
- Jeśli zostanie podana opcja MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, zwracany uchwyt jest uchwytem *współużytkowanym* .

  Zasięg uchwytu współużytkowanego to proces, który jest właścicielem wątku, z którego wywołano wywołanie. Uchwyt może być używany z dowolnego wątku, który należy do tego procesu. Nie wszystkie platformy obsługują wątki.
- Jeśli wywołanie MQCONN lub MQCONNX nie powiedzie się z kodem zakończenia równym MQCC_FAILED, wartość Hconn jest niezdefiniowana.

Tabela 547. Zasięg niewspółużytkowanych uchwytów na różnych platformach	
Platforma	**Zasięg niewspółużytkowanego uchwytu**
z/OS	• CICS: zadanie CICS • IMS: zadanie aż do następnego punktu synchronizacji (z wyłączeniem podzadań zadania) • Zadanie wsadowe z/OS i TSO: zadanie (z wyłączeniem podzadań zadania)
IBM i	Praca
AIX and Linux	Wątek
32-bitowe aplikacje Windows	Wątek
64-bitowe aplikacje Windows	Wątek

W systemie z/OS dla aplikacji CICS zwracana jest wartość:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

CompCode
Typ: MQLONG-wyjście

Szczegółowe informacje zawiera opis parametru **CompCode** w sekcji "MQCONN-Połącz z menedżerem kolejek" na stronie 682 .

Przyczyna
Typ: MQLONG-wyjście

Wywołania MQCONN i MQCONNX mogą zwracać następujące kody. Lista dodatkowych kodów, które mogą być zwracane przez wywołanie MQCONNX, znajduje się w następujących kodach.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

POŁĄCZONO MQRC_ALREADY_CONNECTED
(2002, X'7D2') Aplikacja jest już podłączona.

BŁĄD ŁADOWANIA MQRC_CLUSTER_EXIT_LOAD_ERROR
(2267, X'8DB') Nie można załadować wyjścia obciążenia klastra.

MQRC_SSL_ALREADY_INITIALIZED
(2391, X' 957 ') Protokół SSL został już zainicjowany.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_ADAPTER_CONN_LOAD_ERROR
(2129, X'851 ') Nie można załadować modułu połączenia adaptera.

BŁĄD MQRC_ADAPTER_DEFS_ERROR
(2131, X'853 ') Moduł definicji podsystemu adaptera jest niepoprawny.

MQRC_ADAPTER_DEFS_LOAD_ERROR
(2132, X'854 ') Nie można załadować modułu definicji podsystemu adaptera.

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ADAPTER_STORAGE_NIEDOBÓR
(2127, X'84F') Niewystarczająca ilość pamięci dla adaptera.

MQRC_ANOTHER_Q_MGR_CONNECTED
(2103, X'837 ') Inny menedżer kolejek jest już połączony.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

MQRC_API_EXIT_INIT_BŁĄD
(2375, X' 947 ') Inicjowanie wyjścia funkcji API nie powiodło się.

BŁĄD MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') Zakończenie wyjścia funkcji API nie powiodło się.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ID_KOND_MQRC_W_UŻYCIU
(2160, X'870 ') Identyfikator połączenia jest już używany.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD POŁĄCZENIA MQRC_CONNECTION_
(2273, X'8E1') Błąd podczas przetwarzania wywołania MQCONN.

MQRC_CONNECTION_NOT_AVAILABLE
(2568, X'A08') Występuje w wywołaniu MQCONN lub MQCONNX, gdy menedżer kolejek nie może udostępnić połączenia żądanego typu połączenia w bieżącej instalacji. Nie można nawiązać połączenia z klientem tylko na serwerze. Połączenie lokalne nie może być nawiązywane tylko na kliencie.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CRYPTO_HARDWARE_ERROR
(2382, X'94E') Błąd konfiguracji sprzętu szyfrującego.

MQRC_DUPLICATE_RECOV_COORD
(2163, X'873 ') Istnieje koordynator odtwarzania.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

Dodatkowo w wywołaniu MQCONNX, przekazując blok kontrolny “MQCSP-parametry zabezpieczeń” na stronie 341 z aplikacji CICS lub IMS .

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_HOST_NOT_AVAILABLE (niedostępny)
(2538, X'9EA') Klient wysłał wywołanie MQCONN w celu nawiązania połączenia z menedżerem kolejek, ale próba przydzielenia konwersacji do systemu zdalnego nie powiodła się.

MQRC_INSTALLATION_MISMATCH (Niezgodność instalacji MQ)
(2583, X'A17') Niezgodność między instalacją menedżera kolejek i wybraną biblioteką.

BŁĄD MQRC_KEY_REPOSITORY_ERROR
(2381, X'94D') Repozytorium kluczy jest niepoprawne.

MQRC_MAX_CONNS_LIMIT_REACHED
(2025, X'7E9') Osiągnięto maksymalną liczbę połączeń.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

MQRC_OPEN_FAILED,
(2137, X'859 ') Obiekt nie został pomyślnie otwarty.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_SECURITY_ERROR
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

MQRC_SSL_INITIALIZATION_ERROR (błąd)
(2393, X' 959 ') Błąd inicjowania SSL.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Wywołanie MQCONNX może zwrócić następujące dodatkowe kody przyczyny:

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_AIR_ERROR
(2385, X' 951 ') Niepoprawny rekord informacji uwierzytelniającej.

BŁĄD MQRC_AUTH_INFO_CONN_NAME_ERROR
(2387, X' 953 ') Niepoprawna nazwa połączenia informacji uwierzytelniającej.

BŁĄD MQRC_AUTH_INFO_REC_COUNT_ERROR
(2383, X'94F') Niepoprawna liczba rekordów informacji uwierzytelniającej.

MQRC_AUTH_INFO_REC_BŁĄD
(2384, X' 950 ') Niepoprawne pola rekordu informacji uwierzytelniającej.

BŁĄD MQRC_AUTH_INFO_TYPE_ERROR
(2386, X' 952 ') Niepoprawny typ informacji uwierzytelniającej.

BŁĄD MQRC_CD_ERROR
(2277, X'8E5') Niepoprawna definicja kanału.

MQRC_CLIENT_CONN_BŁĄD
(2278, X'8E6') Niepoprawne pola połączenia klienta.

BŁĄD MQRC_CNO_ERROR
(2139, X'85B') Niepoprawna struktura opcji połączenia.

MQRC_CONN_TAG_IN_UŻYJ
(2271, X'8DF') Znacznik połączenia jest używany.

MQRC_CONN_TAG_NOT_USABLE (nieużywany)
(2350, X'92E') Znacznik połączenia nie nadaje się do użycia.

BŁĄD MQRC_CSP_ERROR
(2595, X'A23') Struktura MQCSP jest niepoprawna.

V 9.3.4 **MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)**
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

BŁĄD MQRC_LDAP_PASSWORD_ERROR
(2390, X' 956 ') Niepoprawne hasło LDAP.

BŁĄD MQRC_LDAP_USER_NAME_ERROR
(2388, X' 954 ') Niepoprawne pola nazwy użytkownika LDAP.

MQRC_LDAP_USER_NAME_LENGTH_ERR
(2389, X' 955 ') Niepoprawna długość nazwy użytkownika LDAP.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD ZASIĘGU MQRC_SCO_ERROR
(2380, X'94C') Niepoprawna struktura opcji konfiguracyjnych SSL.

MQRC_SSL_CONFIG_ERROR
(2392, X' 958 ') Błąd konfiguracji SSL.

MQRC_TOKEN_TIMESTAMP_NOT_VALID,
(2064, X'810 ') Znacznik uwierzytelniania nie jest jeszcze poprawny lub utracił ważność.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

W przypadku języka programowania Visual Basic zastosowanie ma następujący punkt:

- Parametr **ConnectOpts** jest zadeklarowany jako parametr typu MQCNO. Jeśli aplikacja działa jako aplikacja IBM MQ MQI client, a użytkownik chce określić parametry kanału połączenia klienckiego, należy zadeklarować parametr **ConnectOpts** jako typ Any, aby aplikacja mogła określić strukturę MQCNOCD w wywołaniu zamiast struktury MQCNO. Oznacza to jednak, że nie można sprawdzić, czy parametr **ConnectOpts** ma poprawny typ danych.

Wywołanie C

```
MQCONNX (QMgrName, &ConnectOpts, &Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQCONNX' USING QMGRNAME, CONNECTOPTS, HCONN, COMPCODE,
REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQCONNX (QMgrName, ConnectOpts, Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQCONNX,(QMGRNAME,CONNECTOPTS,HCONN,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

```
MQCONNX QMgrName, ConnectOpts, Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-Tworzenie uchwytu komunikatu

Wywołanie MQCRTMH zwraca uchwyt komunikatu.

Aplikacja może używać wywołania MQCRTMH w kolejnych wywołaniach kolejkowania komunikatów:

- Użyj wywołania MQSETMP , aby ustawić właściwość uchwytu komunikatu.
- Za pomocą wywołania MQINQMP można uzyskać informacje o wartości właściwości uchwytu komunikatu.
- Użyj wywołania MQDLTMP , aby usunąć właściwość uchwytu komunikatu.

Uchwyt komunikatu może być używany w wywołaniach MQPUT i MQPUT1 w celu powiązania właściwości uchwytu komunikatu z właściwościami wstawianego komunikatu. Podobnie, określając uchwyt komunikatu w wywołaniu MQGET, można uzyskać dostęp do właściwości pobieranego komunikatu przy użyciu uchwytu komunikatu po zakończeniu wywołania MQGET.

Użyj komendy MQDLTMH , aby usunąć uchwyt komunikatu.

Składnia

MQCRTMH (*Hconn*, *CrtMsgHOpts*, *Hmsg*, *CompCode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX. Jeśli połączenie z menedżerem kolejek przestaje być poprawne i żadne wywołanie IBM MQ nie działa na uchwycie komunikatu, wywołanie MQDLTMH jest niejawnie wywoływane w celu usunięcia komunikatu.

Alternatywnie można podać następującą wartość:

MQHC_UNASSOCIATED_HCONN
Uchwyt połączenia nie reprezentuje połączenia z żadnym konkretnym menedżerem kolejek.

Jeśli ta wartość jest używana, należy usunąć uchwyt komunikatu za pomocą jawnego wywołania komendy MQDLTMH , aby zwolnić przydzieloną do niego pamięć. Program IBM MQ nigdy nie usuwa niejawnie uchwytu komunikatu.

Musi istnieć co najmniej jedno poprawne połączenie z menedżerem kolejek nawiązane w wątku, który utworzył uchwyt komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony błąd MQRC_HCONN_ERROR.

W środowisku z wieloma instalacjami w jednym systemie wartość MQHC_UNASSOCIATED_HCONN jest ograniczona do użycia z pierwszą instalacją załadowaną do procesu. Jeśli uchwyt komunikatu został podany w innej instalacji, zwracany jest kod przyczyny MQRC_HMSG_NOT_AVAILABLE.

W produkcie z/OS dla aplikacji CICS można pominąć wywołanie MQCONN i określić następującą wartość parametru *Hconn*:

MQHC_DEF_KONN
Domyślny uchwyt połączenia

Opcja HOpt CrtMsg
Typ: MQCMHO-wejście

Opcje sterujące działaniem komendy MQCRTMH. Szczegółowe informacje na ten temat zawiera sekcja MQCMHO .

Komunikat Hmsg
Typ: MQHMSG-output

Na wyjściu zwracany jest uchwyt komunikatu, który może być używany do ustawiania, sprawdzania i usuwania właściwości uchwytu komunikatu. Początkowo uchwyt komunikatu nie zawiera żadnych właściwości.

Z uchwytem komunikatu powiązany jest również deskryptor komunikatu. Początkowo zawiera on wartości domyślne. Wartości powiązanych pól deskryptora komunikatu można ustawić i wysłać zapytanie przy użyciu wywołań MQSETMP i MQINQMP. Wywołanie MQDLTMP resetuje pole deskryptora komunikatu do wartości domyślnej.

Jeśli parametr *Hconn* ma wartość MQHC_UNASSOCIATED_HCONN, zwrócony uchwyt komunikatu może być używany w wywołaniach MQGET, MQPUT lub MQPUT1 z dowolnym połączeniem w jednostce przetwarzania, ale jednocześnie może być używany tylko przez jedno wywołanie IBM MQ . Jeśli uchwyt jest używany, gdy drugie wywołanie IBM MQ próbuje użyć tego samego uchwytu komunikatu, drugie wywołanie IBM MQ kończy się niepowodzeniem z kodem przyczyny MQRC_MSG_HANDLE_IN_USE.

Jeśli parametr *Hconn* nie ma wartości MQHC_UNASSOCIATED_HCONN, zwrócony uchwyt komunikatu może być używany tylko w określonym połączeniu.

Ta sama wartość parametru *Hconn* musi być używana w kolejnych wywołaniach MQI, w których używany jest ten uchwyt komunikatu:

- MQDLTMH
- Komenda MQSETMP
- MQINQMP
- Komenda MQDLTMP
- MQMHBUF
- MQBUFMH

Zwrócony uchwyt komunikatu przestaje być poprawny, gdy dla uchwytu komunikatu zostanie wydane wywołanie MQDLTMH lub gdy jednostka przetwarzania definiująca zasięg uchwytu zostanie zakończona. Komenda MQDLTMH jest wywoływana niejawnie, jeśli podczas tworzenia uchwytu komunikatu podano konkretne połączenie, a połączenie z menedżerem kolejek przestaje być poprawne, na przykład w przypadku wywołania MQDBC.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter niedostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

BŁĄD MQRC_CMHO_ERROR
(2461, X'099D') Struktura opcji tworzenia uchwytu komunikatu jest niepoprawna.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2273, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'07E1') Brak dostępnych uchwytów.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Wskaźnik uchwytu komunikatu jest niepoprawny.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

C

```
MQCRTMH (Hconn, &CrtMsgHOpts, &Hmsg, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

kompilatory

```
CALL 'MQCRTMH' USING HCONN, CRTMSGHOPTS, HMSG, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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);
```

Zadeklaruj parametry w następujący sposób:

```
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)
```

Zadeklaruj parametry w następujący sposób:

```
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-wywołania zwrotne sterowania

Wywołanie MQCTL wykonuje działania sterujące na wywołaniach zwrotnych i uchwytach obiektów otwartych dla połączenia.

Składnia

MQCTL (*Hconn*, *Operation*, *ControlOpts*, *CompCode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i określić następującą wartość specjalną dla parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Operacja
Typ: MQLONG-input

Operacja przetwarzana w wywołaniu zwrotnym zdefiniowanym dla określonego uchwytu obiektu. Należy podać jedną i tylko jedną z następujących opcji:

MQOP_START

Rozpocznij odbieranie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Wywołania zwrotne działają w wątku uruchomionym przez system, który jest inny niż każdy wątek aplikacji.

Ta operacja umożliwia sterowanie dostarczonym uchwytem połączenia z systemem. Jedyne wywołania MQI, które mogą być wykonywane przez wątek inny niż wątek konsumenta, to:

- MQCTL z operacją MQOP_STOP
- MQCTL z operacją MQOP_SUSPEND
- MQDISC-wykonuje operację MQCTL z operacją MQOP_STOP przed rozłączeniem połączenia HConn.

Funkcja MQRC_HCONN_ASYNC_ACTIVE jest zwracana, jeśli wywołanie funkcji API IBM MQ zostało wysłane podczas uruchamiania uchwytu połączenia, a wywołanie nie pochodzi z funkcji konsumenta komunikatów.

Jeśli konsument komunikatów zatrzyma połączenie podczas wywołania MQCBCT_START_CALL, wywołanie MQCTL zwróci kod przyczyny niepowodzenia MQRC_CONNECTION_STOPPED.

Można to wykonać w funkcji konsumenta. W przypadku tego samego połączenia, co procedura zwrotna, jej jedynym celem jest anulowanie poprzednio wywołanej operacji MQOP_STOP.

Ta opcja nie jest obsługiwana w następujących środowiskach: CICS w systemie z/OS lub jeśli aplikacja jest powiązana z niewielowątkową biblioteką IBM MQ .

OCZEKIWANIE_NA_URUCHOM_MQOPP
Rozpocznij odbieranie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Konsumenty komunikatów działają w tym samym wątku i sterowanie nie jest zwracane do programu wywołującego MQCTL, dopóki:

- Zwolniona za pomocą operacji MQCTL MQOP_STOP lub MQOP_SUSPEND, lub
- Wszystkie procedury konsumenta zostały wyrejestrowane lub zawieszone.

Jeśli wszyscy konsumenci zostaną wyrejestrowani lub zawieszeni, zostanie wykonana niejawna operacja MQOP_STOP.

Tej opcji nie można używać w procedurze wywołania zwrotnego ani dla bieżącego uchwytu połączenia, ani dla żadnego innego uchwytu połączenia. Próba wywołania spowoduje zwrócenie błędu MQRC_ENVIRONMENT_ERROR.

Jeśli w dowolnym momencie podczas operacji MQOP_START_WAIT nie ma zarejestrowanych, niezawieszonych konsumentów, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_NO_CALLBACKS_ACTIVE.

Jeśli podczas operacji MQOP_START_WAIT połączenie jest zawieszone, wywołanie MQCTL zwraca kod przyczyny ostrzeżenia MQRC_CONNECTION_SUSPENDED; połączenie pozostaje uruchomione.

Aplikacja może wydać komendę MQOP_STOP lub MQOP_RESUME. W tym przypadku operacja MQOP_RESUME jest blokami.

Ta opcja nie jest obsługiwana w kliencie jednowątkowym.

MQOP_STOP
Zatrzymaj odbieranie komunikatów i poczekaj, aż wszyscy konsumenci zakończą swoje operacje, zanim ta opcja zostanie zakończona. Ta operacja zwalnia uchwyt połączenia.

Jeśli ta opcja zostanie wywołana z poziomu procedury zwrotnej, nie będzie obowiązywać do momentu zakończenia procedury. Po zakończeniu procedur konsumenta dla komunikatów, które zostały już odczytane, i po wykonaniu wywołań zatrzymania (na żądanie) procedur zwrotnych, nie są już wywoływane żadne procedury konsumenta komunikatów.

W przypadku wywołania poza procedurą zwrotną sterowanie nie jest zwracane do programu wywołującego, dopóki procedury konsumenta dla komunikatów, które zostały już odczytane, nie zostaną zakończone i po wykonaniu wywołań zwrotnych zatrzymania (na żądanie). Same procedury zwrotne pozostają jednak zarejestrowane.

Ta funkcja nie ma wpływu na komunikaty odczytu z wyprzedzeniem. Należy upewnić się, że konsumenci uruchamiają komendę MQCLOSE (MQCO_QUIESCE) z poziomu funkcji zwrotnej, aby określić, czy są dostępne dalsze komunikaty do dostarczenia.

MQOP_ZAWIEŚ
Wstrzymaj odbieranie komunikatów. Ta operacja zwalnia uchwyt połączenia.

Nie ma to wpływu na odczytywanie komunikatów z wyprzedzeniem dla aplikacji. Jeśli użytkownik zamierza zatrzymać odbieranie komunikatów przez długi czas, należy rozważyć zamknięcie kolejki i ponowne otwarcie jej w przypadku kontynuowania przetwarzania.

Jeśli wywołanie zostanie wykonane z procedury zwrotnej, nie będzie ono obowiązywać do momentu wyjścia z tej procedury. Po zakończeniu bieżącej procedury nie będą wywoływane żadne procedury konsumenta komunikatów.

W przypadku wywołania poza wywołaniem zwrotnym sterowanie nie jest zwracane do programu wywołującego, dopóki bieżąca procedura konsumenta nie zostanie zakończona i nie zostanie wywołana żadna procedura.

MQOP_RESUME
Wznów odbieranie komunikatów.

Ta opcja jest zwykle generowana z głównego wątku aplikacji, ale może być również używana w procedurze zwrotnej w celu anulowania wcześniejszego żądania zawieszenia wydanego w tej samej procedurze.

Jeśli komenda MQOP_RESUME jest używana do wznowienia operacji MQOP_START_WAIT, operacja jest blowana.

ControlOpts
Typ: MQCTLO-wejście

Opcje sterujące działaniem komendy MQCTL

Szczegółowe informacje na temat struktury zawiera sekcja MQCTLO .

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

BŁĄD MQRC_CALLBACK_LINK_ERROR
(2487, X'9B7') Nie można wywołać procedury zwrotnej

MQRC_CALLBACK_NOT_ ZAREJESTROWANE
(2448, X' 990 ') Nie można wyrejestrować, zawiesić lub wznowić, ponieważ nie ma zarejestrowanego wywołania zwrotnego

BŁĄD MQRC_CALLBACK_ROUTINE_ERROR
(2486, X'9B6') W wywołaniu MQOP_REGISTER określono zarówno CallbackFunction , jak i CallbackName .

Albo podano CallbackFunction lub CallbackName , ale nie jest ona zgodna z obecnie zarejestrowaną funkcją zwrotną.

BŁĄD MQRC_CALLBACK_TYPE_ERROR
(2483, X'9B3') Niepoprawne pole typu CallBack.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

BŁĄD MQRC_CBD_ERROR
(2444, X'98C') Blok opcji jest niepoprawny.

BŁĄD MQRC_CBD_OPTIONS_ERROR
(2484, X'9B4') Niepoprawne pole opcji MQCBD.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CORREL_ID_ERROR
(2207, X'89F') Błąd identyfikatora korelacji.

BŁĄD MQRC_ENVIRONMENT_ERROR
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

MQRC_GET_INHIBITED
(2016, X'7E0') Liczba pobrań zablokowana dla kolejki.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Globalny konflikt jednostek pracy.

BŁĄD MQRC_GMO_ERROR
(2186, X'88A') Niepoprawna struktura opcji Get-message.

MQRC_HANDLE_IN_USE_FOR_UOW,
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_INCONSISTENT_BROWSE (przeglądanie niespójne)
(2259, X'8D3') Niespójna specyfikacja przeglądania.

MQRC_INCONSISTENT_UOW,
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_INVALID_MSG_UNDER_CURSOR,
(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

MQRC_LOCAL_UOW_CONFLICT (konflikt uow_rejestru)
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

BŁĄD MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Opcje zgodności są niepoprawne.

MQRC_MAX_MSG_LENGTH_ERROR
(2485, X'9B5') Niepoprawne pole długości MaxMsg

MQRC_MD_BŁĄD
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

MQRC_MODULE_ENTRY_NOT_FOUND
(2497, X'9C1') W module nie można znaleźć określonego punktu wejścia funkcji.

MQRC_MODULE_INVALID
(2496, X'9C0') Moduł został znaleziony, ale ma niepoprawny typ (32 bit/64 bit) lub nie jest poprawną dll.

MQRC_MODULE_NOT_FOUND
(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie ma uprawnień do ładowania.

BŁĄD MQRC_MSG_ID_ERROR
(2206, X'89E') Błąd identyfikatora komunikatu.

BŁĄD MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

MQRC_MSG_TOKEN_ERROR,
(2331, X'91B') Użycie znacznika komunikatu jest niepoprawne.

MQRC_NOT_OPEN_FOR_BROWSE,
(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

MQRC_NOT_OPEN_FOR_INPUT (MQRC_NOT_OPEN_PUT)
(2037, X'7F5') Kolejka nie jest otwarta do wprowadzania.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

BŁĄD OPERACJI MQRC
(2488, X'9B8') Niepoprawny kod operacji w wywołaniu API

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

BŁĄD MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_SIGNAL_OCZEKUJĄCE
(2069, X'815 ') Sygnał zaległości dla tego uchwytu.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

MQRC_SYNCPOINT_NIEDOSTĘPNE
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

MQRC_UOW_NIEDOSTĘPNE
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

BŁĄD INTERVAL_MQRC_WAIT_ERROR
(2090, X'82A') Niepoprawny odstęp czasu oczekiwania w MQGMO.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Podano niewłaściwą wersję MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Procedury zwrotne muszą sprawdzać odpowiedzi ze wszystkich usług, które wywołują, a jeśli podprogram wykryje warunek, którego nie można rozwiązać, musi wydać komendę MQCB MQOP_DEREGISTER, aby zapobiec powtarzającym się wywołaniom procedury zwrotnej.
2. Jeśli używane jest asynchroniczne wykorzystanie w aplikacji, w której menedżer transakcji XA zarządza transakcjami globalnymi, w tym aktualizacjami produktu IBM MQ, należy wziąć pod uwagę następujące dodatkowe kwestie:
   a. Nie można wywołać MQCTL (MQOP_START) dla **HConn**po jego utworzeniu, po wywołaniu funkcji **xa_open**.

      Wynika to z tego, że serwer **HConn** został przyłączony do kontekstu XA i dlatego nie można uzyskać do niego dostępu w oddzielnym wątku lub wątkach, które są używane przez mechanizm asynchronicznego korzystania.
   b. W przypadku wywołania MQCTL (MQOP_START) w tym scenariuszu wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_ASYNC_XA_CONFLICT (2350).
   c. Poprawne jest wywołanie MQCTL (MQOP_START_WAIT) dla **HConn**po jego utworzeniu, po wywołaniu funkcji **xa_open**.

      Jest to spowodowane tym, że ta metoda uruchamiania mechanizmu asynchronicznego pobierania powoduje uruchomienie wszystkich kolejnych wywołań zwrotnych dla **HConn** w wątku, w którym wykonano wywołanie MQCTL. Oznacza to, że połączenie między **HConn** i wątkiem nie zostanie utracone.
3. z/OS W systemie z/OS, gdy operacją jest MQOP_START:
   - Programy, które korzystają z asynchronicznych procedur zwrotnych, muszą mieć uprawnienia do korzystania z języka z/OS UNIX System Services (z/OS UNIX).
   - Programy środowiska językowego (Language Environment-LE), które korzystają z asynchronicznych procedur zwrotnych, muszą korzystać z opcji środowiska wykonawczego środowiska LE POSIX(ON).
   - Programy inne niż LE, które korzystają z asynchronicznych procedur zwrotnych, nie mogą używać interfejsu z/OS UNIX pthread_create (wywoływalna usługa BPX1PTC).
4. z/OS Funkcja MQCTL nie jest obsługiwana w adapterze IMS .

Uwaga: W produkcie CICSopcja MQOP_START nie jest obsługiwana. Zamiast tego należy użyć wywołania funkcji MQOP_START_WAIT.

Wywołanie C

```
MQCTL (Hconn, Operation, &ControlOpts, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQCTL' USING HCONN, OPERATION, CTLOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQCTL(Hconn, Operation, CtlOpts, CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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-Rozłączanie menedżera kolejek

Wywołanie MQDISC przerywa połączenie między menedżerem kolejek a aplikacją i jest odwrotnością wywołania MQCONN lub MQCONNX.

- W systemie z/OSwszystkie aplikacje, które używają asynchronicznego przetwarzania komunikatów, obsługi zdarzeń lub wywołania zwrotnego, główny wątek sterujący musi wywołać wywołanie MQDISC przed zakończeniem. Więcej informacji na ten temat zawiera sekcja Asynchroniczne wykorzystanie komunikatów produktu IBM MQ .
- W systemie z/OSaplikacje produktu CICS nie muszą wywoływać tego wywołania w celu rozłączenia się z menedżerem kolejek.

  Jeśli aplikacja CICS wykonuje to wywołanie, nie ma ono wpływu, chyba że wykonano wcześniejsze wywołanie MQCONNX, podając jedną z następujących wartości:

  MQCNO_SERIALIZE_CONN_TAG_Q_MGR
  MQCNO_SERIALIZE_CONN_TAG_QSG

MQCNO_RESTRICT_CONN_TAG_Q_MGR lub
MQCNO_RESTRICT_CONN_TAG_QSG

opcje, w którym to przypadku wszystkie aktualnie otwarte uchwyty obiektów są zamknięte.

Składnia

MQDISC (*Hconn*, *CompCode*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input/output

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W systemie z/OS dla aplikacji CICS można pominąć wywołanie MQCONN i określić następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia parametr *Hconn* na wartość, która nie jest poprawnym uchwytem dla środowiska. Wartość ta jest następująca:

MQHC_UNUSABLE_HCONN
Bezużyteczny uchwyt połączenia.

W systemie z/OSparametr *Hconn* jest ustawiany na wartość, która nie jest zdefiniowana.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_BACKED_OUT
(2003, X'7D3') Jednostka pracy wycofała się.

MQRC_CONN_TAG_NOT_RELEASED
(2344, X' 928 ') Znacznik połączenia nie został zwolniony.

MQRC_OUTCOME_PENDING (oczekiwanie na wynik MQ)
(2124, X'84C') Wynik operacji zatwierdzania jest w toku.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_ADAPTER_DISC_LOAD_ERROR
(2138, X'85A') Nie można załadować modułu rozłączania adaptera.

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

MQRC_API_EXIT_INIT_BŁĄD
(2375, X' 947 ') Inicjowanie wyjścia funkcji API nie powiodło się.

BŁĄD MQRC_API_EXIT_TERM_ERROR
(2376, X' 948 ') Zakończenie wyjścia funkcji API nie powiodło się.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_OUTCOME_MIXED
(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Jeśli wywołanie MQDISC zostanie wykonane, gdy połączenie nadal ma otwarte obiekty w ramach tego połączenia, menedżer kolejek zamknie te obiekty z opcjami zamknięcia ustawionymi na wartość MQCO_NONE.
2. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, dyspozycja tych zmian zależy od sposobu zakończenia aplikacji:

   a. Jeśli aplikacja wysyła wywołanie MQDISC przed zakończeniem:

   - W przypadku jednostki pracy koordynowanej przez menedżer kolejek menedżer kolejek wysyła wywołanie MQCMIT w imieniu aplikacji. Jednostka pracy jest zatwierdzana, jeśli to możliwe, i wycofywana, jeśli nie.

- W przypadku koordynowanej zewnętrznie jednostki pracy status jednostki pracy nie ulega zmianie. Jednak menedżer kolejek zwykle wskazuje, że jednostka pracy musi zostać zatwierdzona na żądanie koordynatora jednostki pracy.

  W systemach z/OS, CICS, IMS (programy inne niż wsadowe DL/1) i aplikacje RRS są podobne do tych.

b. Jeśli aplikacja zakończy działanie normalnie, ale bez wywołania MQDISC, podjęte działanie zależy od środowiska:

- W systemie z/OS, z wyjątkiem aplikacji MQ Java lub MQ JMS , wykonywane są działania opisane w uwadze 2a .
- We wszystkich innych przypadkach wykonywane są czynności opisane w uwadze 2c .

Ze względu na różnice między środowiskami należy upewnić się, że aplikacje, które mają zostać zatwierdzone, lub wycofać jednostkę pracy, zanim zostaną zakończone.

c. Jeśli aplikacja zakończy działanie *nieprawidłowo* bez wywołania MQDISC, jednostka pracy zostanie wycofana.

3. W systemie z/OSmają zastosowanie następujące punkty:

- Aplikacje CICS nie muszą wywoływać wywołania MQDISC w celu rozłączenia się z menedżerem kolejek, ponieważ sam system CICS nawiązuje połączenie z menedżerem kolejek, a wywołanie MQDISC nie ma wpływu na to połączenie.
- Aplikacje CICS, IMS (inne niż wsadowe programy DL/1) i RRS używają jednostek pracy koordynowanych przez zewnętrznego koordynatora jednostki pracy. W wyniku tego wywołanie MQDISC nie ma wpływu na status jednostki pracy (jeśli istnieje), która istniała w momencie wywołania.

  Jednak wywołanie MQDISC *nie* wskazuje końca użycia znacznika połączenia *ConnTag* , który został powiązany z połączeniem przez wcześniejsze wywołanie MQCONNX wydane przez aplikację. Jeśli podczas wywoływania wywołania MQDISC istnieje aktywna jednostka pracy, która odwołuje się do znacznika połączenia, wywołanie zostanie zakończone z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_CONN_TAG_NOT_RELEASED. Znacznik połączenia nie jest dostępny do ponownego wykorzystania, dopóki zewnętrzny koordynator jednostki pracy nie rozwiąże tej jednostki pracy.

Uwaga: W produkcie CICSopcja MQOP_START nie jest obsługiwana. Zamiast tego należy użyć wywołania funkcji MQOP_START_WAIT.

Wywołanie C

```
MQDISC (&Hconn, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;     /* Connection handle */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Wywołanie COBOL

```
CALL 'MQDISC' USING HCONN, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQDISC (Hconn, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn     fixed bin(31);  /* Connection handle */
dcl CompCode  fixed bin(31);  /* Completion code */
dcl Reason    fixed bin(31);  /* Reason code qualifying CompCode */
```

Wywołanie asemblera systemu System/390

```
CALL MQDISC,(HCONN,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN     DS  F  Connection handle
COMPCODE  DS  F  Completion code
REASON    DS  F  Reason code qualifying COMPCODE
```

Wywołanie Visual Basic

```
MQDISC Hconn, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
Dim Hconn    As Long 'Connection handle'
Dim CompCode As Long 'Completion code'
Dim Reason   As Long 'Reason code qualifying CompCode'
```

MQDLTMH-Usuwanie uchwytu komunikatu

Wywołanie MQDLTMH usuwa uchwyt komunikatu i jest odwrotnością wywołania MQCRTMH.

Składnia

MQDLTMH (*Hconn*, *Hmsg*, *DltMsgHOpts*, *CompCode*, *Przyczyna*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony przy użyciu wywołania MQHC_UNASSOCIATED_HCONN, w wątku usuwającym uchwyt komunikatu musi zostać nawiązane poprawne połączenie. W przeciwnym razie wywołanie zakończy się niepowodzeniem z błędem MQRC_CONNECTION_BROKEN.

Komunikat Hmsg
Typ: MQHMSG-input/output

Jest to uchwyt komunikatu, który ma zostać usunięty. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

Po pomyślnym zakończeniu wywołania uchwyt jest ustawiany na niepoprawną wartość dla środowiska. Wartość ta jest następująca:

MQHM_UNUSABLE_HMSG
Uchwyt komunikatu nie do użycia.

Nie można usunąć uchwytu komunikatu, jeśli inne wywołanie IBM MQ , które przeszło ten sam uchwyt komunikatu, jest w toku.

DltMsgHOpts
Typ: MQDMHO-wejście

Szczegółowe informacje na ten temat zawiera sekcja MQDMHO .

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter niedostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_DMHO_ERROR
(2462, X'099E') Niepoprawna struktura opcji usuwania uchwytu komunikatu.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Wskaźnik uchwytu komunikatu jest niepoprawny.

MQRC_MSG_HANDLE_IN_USE,
(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Wywołanie C

```
MQDLTMH (Hconn, &Hmsg, &DltMsgHOpts, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQDLTMH' USING HCONN, HMSG, DLTMSGHOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**  Connection handle
01   HCONN   PIC S9(9) BINARY.

**  Options that control the action of MQDLTMH
01   DLTMSGHOPTS.
COPY CMQDMHOL.

**  Completion code
01  COMPCODE   PIC S9(9) BINARY.

** Reason code qualifying COMPCODE
01   REASON      PIC S9(9) BINARY.
```

Wywołanie PL/I

```
call MQDLTMH (Hconn, Hmsg, DltMsgHOpts, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
dcl Hconn          /* Connection handle */
dcl Hmsg           /* Message handle */
dcl DltMsgHOpts like MQDMHO;    /* Options that control the action of MQDLTMH */
dcl CompCode       /* Completion code */
dcl Reason         /* Reason code qualifying CompCode */
```

Wywołanie programu High Level Assembler

```
CALL MQDLTMH,(HCONN,HMSG,DLTMSGHOPTS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-właściwość usuwania komunikatu

Wywołanie MQDLTMP usuwa właściwość z uchwytu komunikatu i jest odwrotnością wywołania MQSETMP.

Składnia

MQDLTMP (*Hconn*, *Hmsg*, *DltPropOpts*, *Name*, *CompCode*, *Reason*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony przy użyciu wywołania MQHC_UNASSOCIATED_HCONN, w wątku usuwającym uchwyt komunikatu musi zostać nawiązane poprawne połączenie.
W przeciwnym razie wywołanie nie powiedzie się i zostanie zerwana wartość MQRC_CONNECTION_BROKEN.

Komunikat Hmsg
Typ: MQHMSG-input

Jest to uchwyt komunikatu zawierający właściwość, która ma zostać usunięta. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

DltProp-opcje
Typ: MQDMPO-input

Szczegółowe informacje można znaleźć w opisie typu danych MQDMPO .

Nazwa
Typ: MQCHARV-input

Nazwa właściwości do usunięcia. Więcej informacji na temat nazw właściwości zawiera sekcja Nazwy właściwości .

Znaki wieloznaczne nie są dozwolone w nazwie właściwości.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

WŁAŚCIWOŚĆ_MQRC_NIEDOSTĘPNA
(2471, X'09A7') Właściwość niedostępna.

BŁĄD MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter niedostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'086D') Główne identyfikatory ASID są różne.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_DMPO_ERROR
(2481, X'09B1') Niepoprawna struktura opcji usuwania właściwości komunikatu.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

MQRC_MSG_HANDLE_IN_USE,
(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Niepoprawna nazwa właściwości.

BŁĄD MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Niepoprawny identyfikator kodowanego zestawu znaków nazwy właściwości.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Wywołanie C

```
MQDLTMP (Hconn, Hmsg, &DltPropOpts, &Name, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQDLTMP' USING HCONN, HMSG, DLTPROPOPTS, NAME, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQDLTMP (Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQDLTMP,(HCONN,HMSG,DLTPROPOPTS,NAME,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN        DS         F     Connection handle
HMSG         DS         D     Message handle
DLTPROPOPTS  CMQDMPOA   ,     Options that control the action of MQDLTMP
NAME         CMQCHRVA   ,     Property name
COMPCODE     DS         F     Completion code
REASON       DS         F     Reason code qualifying COMPCODE
```

MQGET-pobierz komunikat

Wywołanie MQGET pobiera komunikat z kolejki lokalnej, która została otwarta za pomocą wywołania MQOPEN.

Składnia

MQGET (*Hconn*, *Hobj*, *MsgDesc*, *GetMsgOpts*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN

Domyślny uchwyt połączenia.

Hobj
Typ: MQHOBJ-input

Ten uchwyt reprezentuje kolejkę, z której ma zostać pobrany komunikat. Wartość *Hobj* została zwrócona przez poprzednie wywołanie MQOPEN. Kolejka musi być otwarta za pomocą co najmniej jednej z następujących opcji (szczegółowe informacje można znaleźć w sekcji "MQOPEN-otwarcie obiektu" na stronie 756):

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE (wyłączna)
- MQOO_INPUT_AS_Q_DEF
- MQOO_BROWSE,

MsgDesc
Typ: MQMD-wejście/wyjście

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu. Szczegółowe informacje można znaleźć w sekcji "MQMD-deskryptor komunikatu" na stronie 432.

Jeśli parametr *BufferLength* jest mniejszy niż długość komunikatu, wartość *MsgDesc* jest wypełniana przez menedżer kolejek, niezależnie od tego, czy parametr **GetMsgOpts** ma wartość MQGMO_ACCEPT_TRUNCATED_MSG (patrz sekcja MQGMO-pole opcji).

Jeśli aplikacja udostępnia deskryptor MQMD version-1 , zwracany komunikat ma przedrostek MQMDE do danych komunikatu aplikacji, ale tylko wtedy, gdy co najmniej jedno z pól w deskryptorze MQMDE ma wartość inną niż domyślna. Jeśli wszystkie pola w MQMDE mają wartości domyślne, MQMDE jest pomijane. Nazwa formatu MQFMT_MD_EXTENSION w polu *Format* w MQMD wskazuje, że istnieje MQMDE.

Aplikacja nie musi udostępniać struktury MQMD, jeśli w polu *MsgHandle* podano poprawny uchwyt komunikatu. Jeśli w tym polu nie zostanie podana żadna wartość, deskryptor komunikatu jest pobierany z deskryptora powiązanego z uchwytami komunikatu.

Jeśli aplikacja udostępnia uchwyt komunikatu, a nie strukturę MQMD i określa właściwość MQGMO_PROPERTIES_FORCE_MQRFH2, wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_MD_ERROR. Wywołanie również kończy się niepowodzeniem z kodem przyczyny MQRC_MD_ERROR, jeśli aplikacja nie udostępnia struktury MQMD i określa opcję MQGMO_PROPERTIES_AS_Q_DEF, a atrybut kolejki **PropertyControl** ma wartość MQPROP_FORCE_MQRFH2.

Jeśli określono opcje dopasowania i używany jest deskryptor komunikatu powiązany z uchwytem komunikatu, pola wejściowe używane do dopasowania pochodzą z uchwytu komunikatu.

Opcja GetMsg
Typ: MQGMO-wejście/wyjście

Szczegółowe informacje można znaleźć w sekcji "MQGMO-opcje pobierania komunikatów" na stronie 376.

BufferLength
Typ: MQLONG-input

Jest to długość (w bajtach) obszaru *Buffer* . Podaj wartość zero dla komunikatów, które nie zawierają danych lub jeśli komunikat ma zostać usunięty z kolejki, a dane usunięte (w tym przypadku należy podać wartość MQGMO_ACCEPT_TRUNCATED_MSG).

Uwaga: Długość najdłuższego komunikatu, jaki można odczytać z kolejki, jest określona przez atrybut kolejki **MaxMsgLength** ; patrz "Atrybuty kolejek" na stronie 863.

Buforuj
Typ: MQBYTExBuffer-długość-dane wyjściowe

Jest to obszar, w którym będą przechowywane dane komunikatu. Wyrównaj bufor na granicy odpowiedniej do rodzaju danych w komunikacie. Wyrównanie 4-bajtowe jest odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówków IBM MQ), ale

niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli wartość parametru *BufferLength* jest mniejsza niż długość komunikatu, do pliku **Buffer**zostanie przeniesiona możliwie jak najwięcej komunikatów. Dzieje się tak, jeśli parametr **GetMsgOpts** ma wartość MQGMO_ACCEPT_TRUNCATED_MSG (więcej informacji na ten temat zawiera sekcja MQGMO-pole opcji).

Zestaw znaków i kodowanie danych w pliku **Buffer** są podawane w polach *CodedCharSetId* i *Encoding* zwracanych w parametrze **MsgDesc** . Jeśli te wartości różnią się od wartości wymaganych przez odbiornik, odbiorca musi dokonać konwersji danych komunikatu aplikacji na zestaw znaków i wymagane kodowanie. Aby przekształcić dane komunikatu, można użyć opcji MQGMO_CONVERT (w razie potrzeby z zapisanym przez użytkownika wyjściem). Szczegółowe informacje na temat tej opcji zawiera sekcja “MQGMO-opcje pobierania komunikatów” na stronie 376 .

Uwaga: Wszystkie pozostałe parametry w wywołaniu MQGET mają zestaw znaków i kodowanie dla lokalnego menedżera kolejek (określone przez atrybut menedżera kolejek **CodedCharSetId** i parametr MQENC_NATIVE).

Jeśli wywołanie nie powiedzie się, zawartość buforu może nadal ulec zmianie.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void: jako parametr można podać adres dowolnego typu danych.

Jeśli parametr **BufferLength** ma wartość zero, nie jest przywoływany parametr *Buffer* ; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość null.

DataLength
Typ: MQLONG-wyjście

Jest to długość (w bajtach) danych aplikacji *w komunikacie*. Jeśli ta wartość jest większa niż *BufferLength*, w parametrze **Buffer** (czyli komunikat jest obcinany) są zwracane tylko *BufferLength* bajtów. Jeśli wartość wynosi zero, komunikat nie zawiera danych aplikacji.

Jeśli wartość parametru *BufferLength* jest mniejsza niż długość komunikatu, parametr *DataLength* jest nadal wypełniany przez menedżer kolejek, niezależnie od tego, czy w parametrze **GetMsgOpts** określono parametr MQGMO_ACCEPT_TRUNCATED_MSG (więcej informacji na ten temat zawiera sekcja MQGMO-pole opcji). Umożliwia to aplikacji określenie wielkości buforu wymaganego do obsługi danych komunikatu, a następnie ponowne wywołanie wywołania z buforem o odpowiedniej wielkości.

Jeśli jednak określono opcję MQGMO_CONVERT, a przekształcone dane komunikatu są zbyt długie, aby zmieścić się w pliku *Buffer*, wartość zwracana dla parametru *DataLength* wynosi:

- Długość *nieprzekształconych* danych dla formatów zdefiniowanych przez menedżera kolejek.

  W takim przypadku, jeśli rodzaj danych powoduje rozszerzenie podczas konwersji, aplikacja musi przydzielić bufor większy niż wartość zwrócona przez menedżer kolejek dla programu *DataLength*.
- Wartość zwracana przez wyjście konwersji danych dla formatów zdefiniowanych przez aplikację.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Wymienione kody przyczyny są tymi, które menedżer kolejek może zwrócić dla parametru **Reason** . Jeśli aplikacja określa opcję MQGMO_CONVERT i zostanie wywołana procedura wyjścia napisana przez użytkownika w celu przekształcenia niektórych lub wszystkich danych komunikatu, wyjście decyduje o tym, jaka wartość jest zwracana dla parametru **Reason** . W wyniku tego możliwe są wartości inne niż podane w dokumentacji.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

MQRC_CONVERTED_STRING_TOO_BIG
(2190, X'88E') Przekształcony łańcuch jest zbyt duży dla pola.

BŁĄD MQRC_DBCS_ERROR
(2150, X'866 ') Niepoprawny łańcuch DBCS.

BŁĄD MQRC_FORMAT_ERROR
(2110, X'83E') Niepoprawny format komunikatu.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

Identyfikatory MQRC_INCONSISTENT_CCSID
(2243, X'8C3') Segmenty komunikatów mają różne identyfikatory CCSID.

MQRC_INCONSISTENT_ENCODINGS,
(2244, X'8C4') Segmenty komunikatów mają różne kodowania.

MQRC_INCONSISTENT_UOW,
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_MSG_TOKEN_ERROR,
(2331, X'91B') Niepoprawne użycie znacznika komunikatu.

MQRC_NO_MSG_LOCKED,
(2209, X'8A1') Brak zablokowanego komunikatu.

MQRC_NOT_CONVERTED (nie skonwertowany)
(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

MQRC_OPTIONS_CHANGED
(nnnn, X'xxx ') Zmieniono opcje, które były wymagane do zachowania spójności.

MQRC_PARTIALLY_CONVERTED
(2272, X'8E0') Dane komunikatu zostały częściowo przekształcone.

MQRC_SIGNAL_REQUEST_ACCEPTED
(2070, X'816 ') Nie został zwrócony żaden komunikat (ale żądanie sygnału zostało zaakceptowane).

BŁĄD MQRC_SOURCE_BUFFER_ERROR
(2145, X'861 ') Niepoprawny parametr buforu źródłowego.

BŁĄD MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Niepoprawny identyfikator źródłowego kodowanego zestawu znaków.

BŁĄD MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841 ') Nie rozpoznano kodowania dziesiętnego w komunikacie.

MQRC_SOURCE_FLOAT_ENC_ERROR.
(2114, X'842 ') Nierozpoznane kodowanie zmiennopozycyjne w komunikacie.

BŁĄD MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840 ') Nierozpoznane kodowanie źródłowe liczby całkowitej.

BŁĄD MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Niepoprawny parametr długości źródła.

BŁĄD MQRC_TARGET_BUFFER_ERROR
(2146, X'862 ') Niepoprawny parametr buforu docelowego.

BŁĄD MQRC_TARGET_CCSID_ERROR
(2115, X'843 ') Niepoprawny identyfikator kodowanego zestawu znaków.

BŁĄD MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845 ') Nierozpoznane kodowanie Packed-decimal określone przez odbiornik.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846 ') Nierozpoznane kodowanie zmiennopozycyjne określone przez odbiornik.

BŁĄD MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844 ') Nie rozpoznano kodowania docelowej liczby całkowitej.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Zwrócono obcięty komunikat (przetwarzanie zakończone).

Niepowodzenie MQRC_TRUNCATED_MSG_FAILED
(2080, X'820 ') Zwrócono obcięty komunikat (przetwarzanie nie zostało zakończone).

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

MQRC_ADAPTER_CONV_LOAD_ERROR
(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BACKED_OUT
(2003, X'7D3') Jednostka pracy wycofała się.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'7D4') Niepoprawny parametr buforu.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_NIEDOSTĘPNE
(2345, X' 929 ') Obiekt sprzęgania nie jest dostępny.

MQRC_CF_STRUC_FAILED
(2373, X' 945 ') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Nagłówek listy struktury narzędzia CF w użyciu.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CORREL_ID_ERROR
(2207, X'89F') Błąd identyfikatora korelacji.

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Niepoprawny parametr długości danych.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

MQRC_GET_INHIBITED
(2016, X'7E0') Liczba pobrań zablokowana dla kolejki.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Globalny konflikt jednostek pracy.

BŁĄD MQRC_GMO_ERROR
(2186, X'88A') Niepoprawna struktura opcji Get-message.

MQRC_HANDLE_IN_USE_FOR_UOW,
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_INCONSISTENT_BROWSE (przeglądanie niespójne)
(2259, X'8D3') Niespójna specyfikacja przeglądania.

MQRC_INCONSISTENT_UOW,
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_INVALID_MSG_UNDER_CURSOR,
(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

MQRC_LOCAL_UOW_CONFLICT (konflikt uow_rejestru)
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

BŁĄD MQRC_MATCH_OPTIONS_ERROR
(2247, X'8C7') Opcje zgodności są niepoprawne.

MQRC_MD_BŁĄD
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

BŁĄD MQRC_MSG_ID_ERROR
(2206, X'89E') Błąd identyfikatora komunikatu.

BŁĄD MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

MQRC_MSG_TOKEN_ERROR,
(2331, X'91B') Użycie znacznika komunikatu jest niepoprawne.

MQRC_NO_MSG_AVAILABLE
(2033, X'7F1') Brak dostępnego komunikatu.

MQRC_NO_MSG_UNDER_CURSOR,
(2034, X'7F2') Kursor przeglądania nie jest ustawiony na komunikacie.

MQRC_NOT_OPEN_FOR_BROWSE,
(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

MQRC_NOT_OPEN_FOR_INPUT (MQRC_NOT_OPEN_PUT)
(2037, X'7F5') Kolejka nie jest otwarta do wprowadzania.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

BŁĄD MQRC_Q_INDEX_TYPE_ERROR
(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

MQRC_SECOND_MARK_NOT_ALLOWED
(2062, X'80E') Komunikat jest już oznaczony.

MQRC_SIGNAL_OCZEKUJĄCE
(2069, X'815 ') Sygnał zaległości dla tego uchwytu.

MQRC_SIGNAL1_ERROR
(2099, X'833 ') Niepoprawne pole sygnału.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

MQRC_SYNCPOINT_NIEDOSTĘPNE
(2072, X'818 ') Obsługa punktu synchronizacji jest niedostępna.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

MQRC_UOW_NIEDOSTĘPNE
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

BŁĄD INTERVAL_MQRC_WAIT_ERROR
(2090, X'82A') Niepoprawny odstęp czasu oczekiwania w MQGMO.

MQRC_WRONG_GMO_VERSION
(2256, X'8D0') Podano niewłaściwą wersję MQGMO.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Pobrany komunikat jest zwykle usuwany z kolejki. To usunięcie może nastąpić w ramach samego wywołania MQGET lub w ramach punktu synchronizacji.

   Dostępne są następujące opcje przeglądania: MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT i MQGMO_BROWSE_MSG_UNDER_CURSOR.

2. Jeśli opcja MQGMO_LOCK jest określona z jedną z opcji przeglądania, przeglądany komunikat jest zablokowany, aby był widoczny tylko dla tego uchwytu.

   Jeśli określono opcję MQGMO_UNLOCK, odblokowywany wcześniej komunikat jest odblokowywany. W tym przypadku nie jest pobierany żaden komunikat, a parametry **MsgDesc**, **BufferLength**, **Buffer**i **DataLength** nie są sprawdzane ani zmieniane.

3. W przypadku aplikacji wysyłających wywołanie MQGET pobrany komunikat może zostać utracony, jeśli aplikacja zakończy działanie nieprawidłowo lub połączenie zostanie przerwane podczas przetwarzania wywołania. Ten problem występuje, ponieważ odpowiednik działający na tej samej platformie co menedżer kolejek, który wywołuje wywołanie MQGET w imieniu aplikacji, nie może wykryć utraty aplikacji, dopóki odpowiednik nie zwróci komunikatu do aplikacji po usunięciu komunikatu z kolejki. Ten problem może wystąpić zarówno w przypadku komunikatów trwałych, jak i nietrwałych.

   Aby wyeliminować ryzyko utraty komunikatów w ten sposób, zawsze pobieraj komunikaty w obrębie jednostek pracy. Oznacza to, że przez określenie opcji MQGMO_SYNCPOINT w wywołaniu MQGET i użycie wywołań MQCMIT lub MQBACK w celu zatwierdzenia lub wycofania jednostki pracy po zakończeniu przetwarzania komunikatów. Jeśli określono parametr MQGMO_SYNCPOINT i klient kończy działanie nieprawidłowo lub połączenie jest zerwane, odpowiednik wycofuje jednostkę pracy w menedżerze kolejek i komunikat jest przywracany do kolejki. Więcej informacji na temat punktów synchronizacji zawiera sekcja Uwagi dotyczące punktów synchronizacji w aplikacjach IBM MQ.

   Taka sytuacja może wystąpić w przypadku klientów produktu IBM MQ oraz aplikacji działających na tej samej platformie co menedżer kolejek.

4. Jeśli aplikacja umieszcza sekwencję komunikatów w określonymw obrębie pojedynczej jednostki pracy, a następnie pomyślnie zatwierdza tę jednostkę pracy, komunikaty stają się dostępne do pobrania w następujący sposób:

   - Jeśli kolejka jest *kolejką niewspółużytkowaną* (czyli kolejką lokalną), wszystkie komunikaty w jednostce pracy stają się dostępne w tym samym czasie.
   - Jeśli kolejka jest *kolejką współużytkowaną*, komunikaty w jednostce pracy stają się dostępne w kolejności, w jakiej zostały umieszczone, ale nie wszystkie w tym samym czasie. Jeśli system jest obciążony, możliwe jest pomyślne pobranie pierwszego komunikatu w jednostce pracy, ale wywołanie MQGET dla drugiego lub kolejnego komunikatu w jednostce pracy nie powiedzie się i zostanie zwrócony komunikat MQRC_NO_MSG_AVAILABLE. Jeśli wystąpi ten problem, aplikacja musi odczekać krótki czas, a następnie ponowić operację.

5. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez użycia grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są pewne warunki. Szczegółowe informacje na ten temat zawiera sekcja Uwagi dotyczące użycia MQPUT . Jeśli warunki są spełnione, komunikaty są przedstawiane aplikacji odbierającej w kolejności, w jakiej zostały wysłane, jeśli:

   - Tylko jeden odbiornik otrzymuje komunikaty z kolejki.

     Jeśli istnieją co najmniej dwie aplikacje pobierające komunikaty z kolejki, muszą one uzgodnić z nadawcą mechanizm, który ma być używany do identyfikowania komunikatów należących

do sekwencji. Na przykład nadawca może ustawić wszystkie pola `CorrelId` w komunikatach w sekwencji na wartość unikalną dla tej sekwencji komunikatów.

- Odbiorca nie zmienia celowo kolejności pobierania, na przykład podając konkretną wartość `MsgId` lub `CorrelId`.

Jeśli aplikacja wysyłająca umieszcza komunikaty w grupie komunikatów, komunikaty są prezentowane aplikacji odbierającej w poprawnej kolejności, jeśli aplikacja odbierająca określa opcję MQGMO_LOGICAL_ORDER w wywołaniu MQGET. Więcej informacji na temat grup komunikatów zawiera sekcja:

- MQMD-pole MsgFlags
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER (MQGMO_LOGICAL_ORDER)

Jeśli użytkownik otrzymuje komunikaty w grupie w punkcie synchronizacji, musi upewnić się, że przed podjęciem próby zakończenia transakcji przetwarzana jest pełna grupa.

6. Aplikacje muszą sprawdzić kod sprzężenia zwrotnego MQFB_QUIT w polu `Feedback` parametru **MsgDesc** i zakończyć działanie, jeśli znajdą tę wartość. Więcej informacji na ten temat zawiera sekcja MQMD-Feedback field .
7. Jeśli kolejka identyfikowana przez `Hobj` została otwarta za pomocą opcji MQOO_SAVE_ALL_CONTEXT, a kod zakończenia z wywołania MQGET to MQCC_OK lub MQCC_WARNING, kontekst powiązany z uchwytem kolejki `Hobj` jest ustawiony na kontekst komunikatu, który został pobrany (chyba że opcje MQGMO_BROWSE_FIRST, MQGMO_BROWSE_NEXT lub MQGMO_BROWSE_MSG_UNDER_CURSOR są oznaczone jako niedostępne).

   Zapisanego kontekstu można użyć w kolejnym wywołaniu MQPUT lub MQPUT1 , podając opcje MQPMO_PASS_IDENTITY_CONTEXT lub MQPMO_PASS_ALL_CONTEXT. Umożliwia to przesyłanie kontekstu odebranego komunikatu w całości lub w części do innego komunikatu (na przykład, gdy komunikat jest przekazywany do innej kolejki). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.
8. Jeśli w parametrze **GetMsgOpts** zostanie podana opcja MQGMO_CONVERT, dane komunikatu aplikacji są przekształcane w reprezentację żądaną przez aplikację odbierającą przed umieszczeniem danych w parametrze **Buffer** :

   - Pole `Format` w informacjach sterujących komunikatu identyfikuje strukturę danych aplikacji, a pola `CodedCharSetId` i `Encoding` w informacjach sterujących komunikatu określają jego identyfikator zestawu znaków i kodowanie.
   - Aplikacja wywołująca wywołanie MQGET określa w polach `CodedCharSetId` i `Encoding` parametru **MsgDesc** identyfikator zestawu znaków i kodowanie, na które mają zostać przekształcone dane komunikatu aplikacji.

   Jeśli konwersja danych komunikatu jest konieczna, konwersja jest wykonywana przez sam menedżer kolejek lub przez wyjście zapisane przez użytkownika, w zależności od wartości pola `Format` w informacjach sterujących komunikatu:

   - Następujące nazwy formatów są formatami, które są przekształcane przez menedżer kolejek; te formaty są nazywane formatami "wbudowanymi":
     - MQFMT_ADMIN,
     - MQFMT_CICS (tylko system z/OS)
     - MQFMT_COMMAND_1
     - MQFMT_COMMAND_2
     - MQFMT_DEAD_LETTER_HEADER
     - Usługa MQFMT_DIST_HEADER
     - MQFMT_EVENT, wersja 1
     - MQFMT_EVENT, wersja 2 (tylko z/OS)

  - MQFMT_IMS,
  - MQFMT_IMS_VAR_STRING (łańcuch zmiennych)
  - MQFMT_MD_EXTENSION
  - MQFMT_PCF,
  - MQFMT_REF_MSG_HEADER
  - ZMQFMT_RF_HEADER
  - MQFMT_RF_HEADER_2
  - MQFMT_STRING
  - MQFMT_TRIGGER,
  - MQFMT_WORK_INFO_HEADER (tylko z/OS)
  - MQFMT_XMIT_Q_HEADER
- Nazwa formatu MQFMT_NONE jest wartością specjalną, która wskazuje, że rodzaj danych w komunikacie jest niezdefiniowany. W związku z tym menedżer kolejek nie podejmuje próby konwersji podczas pobierania komunikatu z kolejki.

  Uwaga: Jeśli w wywołaniu MQGET określono opcję MQGMO_CONVERT dla komunikatu o nazwie formatu MQFMT_NONE, a zestaw znaków lub kodowanie komunikatu różnią się od podanego w parametrze **MsgDesc** , komunikat jest zwracany w parametrze **Buffer** (przy założeniu braku innych błędów), ale wywołanie kończy się z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_FORMAT_ERROR.

  Parametru MQFMT_NONE można użyć, gdy rodzaj danych komunikatu oznacza, że nie wymaga on konwersji lub gdy aplikacje wysyłające i odbierające uzgodniły między sobą formularz, w którym mają zostać wysłane dane komunikatu.
- Wszystkie inne nazwy formatów przekazują komunikat do wyjścia napisanego przez użytkownika w celu konwersji. Wyjście ma taką samą nazwę jak format, z wyjątkiem dodatków specyficznych dla środowiska. Nazwy formatów określone przez użytkownika nie mogą zaczynać się od liter IBM MQ.

  Szczegółowe informacje na temat wyjścia konwersji danych zawiera sekcja “Wyjście konwersji danych” na stronie 937 .

Dane użytkownika w komunikacie mogą być konwertowane między dowolnymi obsługiwanymi zestawami znaków i kodowaniami. Należy jednak pamiętać, że jeśli komunikat zawiera jedną lub więcej struktur nagłówka IBM MQ , nie można dokonać konwersji komunikatu z lub na zestaw znaków, który zawiera znaki dwubajtowe lub wielobajtowe dla żadnego ze znaków poprawnych w nazwach kolejek. Kod przyczyny MQRC_SOURCE_CCSID_ERROR lub MQRC_TARGET_CCSID_ERROR jest zwracany w przypadku próby wykonania tej operacji, a komunikat jest zwracany bez konwersji. Przykładem takiego zestawu znaków jest zestaw znaków Unicode UTF-16 .

W przypadku powrotu z operacji MQGET następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:

- MQRC_BRAK

Następujący kod przyczyny wskazuje, że komunikat mógł zostać pomyślnie przekształcony. Aby uzyskać więcej informacji, aplikacja musi sprawdzić pola `CodedCharSetId` i `Encoding` w parametrze **MsgDesc** :

- MQRC_TRUNCATED_MSG_ACCEPTED

Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Uwaga: Interpretacja tego kodu przyczyny jest prawdziwa w przypadku konwersji wykonywanych przez program zewnętrzny napisany przez użytkownika tylko wtedy, gdy program zewnętrzny jest zgodny z wytycznymi przetwarzania opisanymi w sekcji “Wyjście konwersji danych” na stronie 937.

9. Jeśli do pobierania komunikatów używany jest interfejs obiektowy, można nie określać buforu do przechowywania danych komunikatu dla wywołania MQGET. Jeśli komunikat jest otrzymany za pomocą aplikacji obiektowej bez ograniczenia wielkości buforu komunikatów odbiorczych, aplikacja

nie kończy się niepowodzeniem z komunikatem MQRC_CONVERTED_MSG_TOO_BIG i odbiera przekształcony komunikat. Dotyczy to następujących środowisk:

- .NET, w tym w pełni zarządzane aplikacje
- C++
- Java (IBM MQ classes for Java)

Uwaga: Dla wszystkich klientów, jeśli wartość `sharingConversations` wynosi zero i jeśli bufor jest zbyt mały, aby odebrać przekształcony komunikat, zwracany jest nieprzekształcony komunikat z kodem przyczyny MQRC_CONVERTED_MSG_TOO_BIG. Więcej informacji na temat produktu `sharingConversations`zawiera sekcja Używanie współużytkowania konwersacji w aplikacji klienckiej.

10. W przypadku formatów wbudowanych menedżer kolejek może wykonać *domyślną konwersję* łańcuchów znaków w komunikacie, jeśli określono opcję MQGMO_CONVERT. Konwersja domyślna umożliwia menedżerowi kolejek użycie określonego przez instalację domyślnego zestawu znaków, który przybliża rzeczywisty zestaw znaków podczas konwersji danych łańcuchowych. W rezultacie wywołanie MQGET może zakończyć się pomyślnie z kodem zakończenia MQCC_OK, zamiast z kodem MQCC_WARNING i kodem przyczyny MQRC_SOURCE_CCSID_ERROR lub MQRC_TARGET_CCSID_ERROR.

    Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji danych łańcuchowych jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Aby tego uniknąć, należy używać znaków w łańcuchu, które są wspólne zarówno dla rzeczywistego zestawu znaków, jak i dla domyślnego zestawu znaków.

    Konwersja domyślna dotyczy zarówno danych komunikatu aplikacji, jak i pól znakowych w strukturach MQMD i MQMDE:

    - Domyślna konwersja danych komunikatu aplikacji ma miejsce tylko wtedy, gdy wszystkie poniższe instrukcje są prawdziwe:
        - Aplikacja określa opcję MQGMO_CONVERT.
        - Komunikat zawiera dane, które muszą zostać przekształcone z lub na zestaw znaków, który nie jest obsługiwany.
        - Konwersja domyślna została włączona podczas instalowania lub restartowania menedżera kolejek.
    - Domyślna konwersja pól znakowych w strukturach MQMD i MQMDE jest wykonywana w razie potrzeby, jeśli dla menedżera kolejek włączono konwersję domyślną. Konwersja jest wykonywana nawet wtedy, gdy opcja MQGMO_CONVERT nie jest określona przez aplikację w wywołaniu MQGET.
11. W przypadku języka programowania Visual Basic mają zastosowanie następujące punkty:

    - Jeśli wielkość parametru **`Buffer`** jest mniejsza niż długość określona przez parametr **`BufferLength`** , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_STORAGE_NOT_AVAILABLE.
    - Parametr **`Buffer`** jest zadeklarowany jako typu `String`. Jeśli dane do pobrania z kolejki nie są typu `String`, należy użyćWywołanie MQGETAny zamiast wywołania MQGET.

      Wywołanie MQGETAny ma takie same parametry jak wywołanie MQGET, z tym wyjątkiem, że parametr **`Buffer`** jest zadeklarowany jako typ `Any`, co umożliwia pobranie dowolnego typu danych. Oznacza to jednak, że nie można sprawdzić, czy wielkość pliku `Buffer` wynosi co najmniej `BufferLength` bajtów.
12. Nie wszystkie opcje MQGET są obsługiwane, gdy włączona jest funkcja odczytu z wyprzedzeniem. Poniższa tabela wskazuje, które opcje są dozwolone i czy można je zmieniać między wywołaniami MQGET.

Tabela 548. Opcje MQGET są dozwolone, gdy włączona jest funkcja odczytu z wyprzedzeniem

	Dozwolone, gdy odczyt z wyprzedzeniem jest włączony i można go zmieniać między wywołaniami MQGET	**Dozwolone, gdy odczyt z wyprzedzeniem jest włączony, ale nie można go zmienić między wywołaniami MQGET** [a]	**Opcje MQGET, które nie są dozwolone, gdy włączony jest odczyt z wyprzedzeniem** [b]
Wartości deskryptora MQGET MD	MsgId [c] CorrelId [k]	Kodowanie CodedCharSetId	

Tabela 548. Opcje MQGET są dozwolone, gdy włączona jest funkcja odczytu z wyprzedzeniem (kontynuacja)			
	Dozwolone, gdy odczyt z wyprzedzeniem jest włączony i można go zmieniać między wywołaniami MQGET	**Dozwolone, gdy odczyt z wyprzedzeniem jest włączony, ale nie można go zmienić między wywołaniami MQGET [a]**	**Opcje MQGET, które nie są dozwolone, gdy włączony jest odczyt z wyprzedzeniem [b]**
Opcje 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 (KOLEJNA_KOLEJNA_STRUKTURA) 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] BLOKADA MQGMO_LOCK MQGMO_UNLOCK (odblokowanie MQGMO)
Wartości MQGMO		MsgHandle	

a. Jeśli te opcje zostaną zmienione między wywołaniami MQGET, zostanie zwrócony kod przyczyny MQRC_OPTIONS_CHANGED.

b. Jeśli te opcje zostaną podane podczas pierwszego wywołania MQGET, odczyt z wyprzedzeniem zostanie wyłączony. Jeśli te opcje zostaną podane w kolejnym wywołaniu MQGET, zostanie zwrócony kod przyczyny MQRC_OPTIONS_ERROR.

c. Aplikacje klienckie muszą uwzględniać fakt, że jeśli wartości MsgId i CorrelId zostały zmienione między wywołaniami MQGET, komunikaty z poprzednimi wartościami mogły już zostać wysłane do klienta i pozostają w buforze odczytu z wyprzedzeniem na kliencie, dopóki nie zostaną wykorzystane (lub automatycznie usunięte).

d. Pierwsze wywołanie MQGET określa, czy komunikaty mają być przeglądane lub pobierane z kolejki, gdy włączony jest odczyt z wyprzedzeniem. Jeśli w aplikacji zostanie podjęta próba użycia zarówno operacji przeglądania, jak i pobierania, zostanie zwrócony kod przyczyny MQRC_OPTIONS_CHANGED.

e. Opcja MQGMO_MSG_UNDER_CURSOR nie jest dostępna, jeśli włączony jest odczyt z wyprzedzeniem. Komunikaty można przeglądać albo odbierać, gdy włączony jest odczyt z wyprzedzeniem. Nie można jednak jednocześnie korzystać z obu tych funkcji.

13. Aplikacje mogą destrukcyjnie pobrać niezatwierdzone komunikaty tylko wtedy, gdy są one umieszczane w tej samej lokalnej jednostce pracy, co operacja pobierania. Aplikacje nie mogą uzyskać niezatwierdzonych komunikatów bez zniszczenia.
14. Komunikaty znajdujące się pod kursorem przeglądania mogą być pobierane w jednostce pracy. W ten sposób nie można pobrać niezatwierdzonego komunikatu.

Wywołanie C

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

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQGET' USING HCONN, HOBJ, MSGDESC, GETMSGOPTS, BUFFERLENGTH,
BUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

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

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
        CALL MQGET,(HCONN,HOBJ,MSGDESC,GETMSGOPTS,BUFFERLENGTH,
              BUFFER,DATALENGTH,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

```
MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer,
DataLength, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-zapytanie o obiekt-atrybuty

Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierający atrybuty obiektu.

Poprawne są następujące typy obiektów:

- Menedżer kolejek
- Kolejka
- Lista nazw
- Definicja procesu

Składnia

MQINQ (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *CompCode*, *Reason*)

Parametry

hconn
Typ: MQHCONN -wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX .

W systemie z/OS dla aplikacji CICS można pominąć wywołanie MQCONN i podać następującą wartość dla parametru *Hconn*:

MQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Hobj
Typ: MQHOBJ -wejście

Ten uchwyt reprezentuje obiekt (dowolnego typu) z wymaganymi atrybutami. Uchwyt musi zostać zwrócony przez poprzednie wywołanie MQOPEN , które określało opcję MQOO_INQUIRE .

SelectorCount
Typ: MQLONG -wejście

Jest to liczba selektorów, które są dostarczane w tablicy *Selectors* . Jest to liczba atrybutów, które mają zostać zwrócone. Zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

Selektory
Typ: MQLONG x *SelectorCount* -wejście

Jest to tablica selektorów atrybutów **SelectorCount** . Każdy selektor identyfikuje atrybut (liczbę całkowitą lub znak) z wymaganą wartością.

Każdy selektor musi być poprawny dla typu obiektu reprezentowanego przez *Hobj* , w przeciwnym razie wywołanie zakończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_SELECTOR_ERROR.

W przypadku kolejek specjalnych:

- Jeśli selektor nie jest poprawny dla kolejek dowolnego typu, wywołanie kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_SELECTOR_ERROR.
- Jeśli selektor ma zastosowanie tylko do kolejek innych typów niż typ obiektu, wywołanie powiedzie się z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_SELECTOR_NOT_FOR_TYPE.
- Jeśli kolejka, której dotyczy zapytanie, jest kolejką klastra, poprawne selektory zależą od sposobu rozstrzygnięcia kolejki. Więcej informacji na ten temat zawiera sekcja “Użycie notatek” na stronie 743 .

Selektory można określić w dowolnej kolejności. Wartości atrybutów odpowiadające selektorom atrybutów będącym liczbami całkowitymi (MQIA_* selektory) są zwracane w produkcie *IntAttrs* w tej samej kolejności, w jakiej selektory te występują w produkcie *Selectors*. Wartości atrybutów odpowiadające selektorom atrybutów znakowych (selektorom MQCA_*) są zwracane w produkcie *CharAttrs* w tej samej kolejności, w jakiej występują te selektory. Selektory MQIA_* mogą być przeplatane z selektorami MQCA_* ; istotna jest tylko kolejność względna w obrębie każdego typu.

Uwaga:

1. Selektory atrybutów całkowitych i znakowych są przydzielane w dwóch różnych zakresach: selektory MQIA_* rezydują w zakresie od MQIA_FIRST do MQIA_LASToraz selektory MQCA_* w zakresie od MQCA_FIRST do MQCA_LAST.

   Dla każdego zakresu stałe MQIA_LAST_USED i MQCA_LAST_USED definiują najwyższą wartość akceptowaną przez menedżer kolejek.
2. Jeśli najpierw wystąpią wszystkie selektory MQIA_* , można użyć tych samych numerów elementów do adresowania odpowiednich elementów w tablicach *Selectors* i *IntAttrs* .
3. Jeśli parametr **SelectorCount** ma wartość zero, oznacza to, że nie występuje odwołanie do parametru *Selectors* . W takim przypadku adres parametru przekazany przez programy napisane w języku C lub S/390 może mieć wartość NULL.

Atrybuty, do których można utworzyć zapytanie, są wymienione w poniższych tabelach. For the MQCA_* selectors, the constant that defines the length in bytes of the resulting string in *CharAttrs* is provided in parentheses.

W poniższych tabelach przedstawiono selektory według obiektów w porządku alfabetycznym w następujący sposób:

- Selektory atrybutów produktu Tabela 549 na stronie 729 MQINQ dla kolejek
- Selektory atrybutów produktu Tabela 550 na stronie 731 MQINQ dla list nazw
- Selektory atrybutów produktu Tabela 551 na stronie 732 MQINQ dla definicji procesów
- Selektory atrybutów produktu Tabela 552 na stronie 732 MQINQ dla menedżera kolejek

Wszystkie selektory są obsługiwane na wszystkich platformach IBM MQ , z wyjątkiem wskazanych w kolumnie **Uwaga** w następujący sposób:

NIEz/OS
Obsługiwane na wszystkich platformach **z wyjątkiem** z/OS

z/OS
Obsługiwane **tylko** w systemie z/OS

Tabela 549. Selektory atrybutów MQINQ dla kolejek

Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Czas ostatniej zmiany	
MQCA_BACKOUT_REQ_Q_NAME	MQ_Q_NAME_LENGTH	Nadmierna liczba wycofanych komunikatów o nazwie	
MQCA_BASE_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki, na którą alias jest tłumaczony	
MQCA_CF_STRUC_NAME	MQ_CF_STRUC_NAME_LENGTH	Nazwa struktury narzędzia CF	z/OS
MQCA_CLUS_CHL_NAME	MQ_CHANNEL_NAME_LENGTH	Nazwa kanału nadawczego klastra, który używa tej kolejki jako kolejki transmisji.	
MQCA_CLUSTER_NAME	MQ_CLUSTER_NAME_LENGTH	Nazwa klastra	
MQCA_CLUSTER_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Lista nazw klastrów	
MQCA_CREATION_DATE	MQ_CREATION_DATE_LENGTH	Data utworzenia kolejki	
MQCA_CREATION_TIME	MQ_CREATION_TIME_LENGTH	Czas utworzenia kolejki	
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	Atrybut niestandardowy dla nowych funkcji	
MQCA_INITIATION_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki inicjacji	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nazwa definicji procesu	
MQCA_Q_DESC	MQ_Q_DESC_LENGTH	Opis kolejki	
MQCA_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki	
MQCA_REMOTE_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nazwa zdalnego menedżera kolejek	
MQCA_REMOTE_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki zdalnej znana w zdalnym menedżerze kolejek	
MQCA_STORAGE_CLASS	MQ_STORAGE_CLASS_LENGTH	Nazwa klasy pamięci masowej	z/OS
MQCA_TRIGGER_DATA	MQ_TRIGGER_DATA_LENGTH	Dane wyzwalacza	
MQCA_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki transmisji	
MQIA_ACCOUNTING_Q	MQLONG	Steruje kolekcjonowaniem danych rozliczeniowych dla kolejki	NIEz/OS
MQIA_BACKOUT_THRESHOLD	MQLONG	Próg wycofania	
MQIA_CLWL_Q_PRIORITY	MQLONG	Priorytet kolejki	
MQIA_CLWL_Q_RANK	MQLONG	Klasyfikacja kolejki	

Tabela 549. Selektory atrybutów MQINQ dla kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQIA_CLWL_USEQ	MQLONG	Użyj kolejek zdalnych	
MQIA_CURRENT_Q_DEPTH	MQLONG	Liczba komunikatów w kolejce	
MQIA_DEF_BIND	MQLONG	Domyślne łączenie	
MQIA_DEF_INPUT_OPEN_OPTION	MQLONG	Domyślna opcja open-for-input	
MQIA_DEF_PERSISTENCE	MQLONG	Domyślna trwałość komunikatu	
MQIA_DEF_PRIORITY	MQLONG	Domyślny priorytet komunikatu	
MQIA_DEFINITION_TYPE	MQLONG	Typ definicji kolejki.	
MQIA_DIST_LISTS	MQLONG	Obsługa listy dystrybucyjnej	NIEz/OS
MQIA_HARDEN_GET_BACKOUT	MQLONG	Informacja o tym, czy ma być harden backout count	
MQIA_INDEX_TYPE	MQLONG	Typ indeksu obsługiwanego dla kolejki	z/OS
MQIA_INHIBIT_GET	MQLONG	Określa, czy operacje pobierania są dozwolone	
MQIA_INHIBIT_PUT	MQLONG	Określa, czy operacje umieszczania (put) są dozwolone	
MQIA_MAX_MSG_LENGTH	MQLONG	Maksymalna długość komunikatu	
MQIA_MAX_Q_DEPTH	MQLONG	Maksymalna liczba komunikatów dozwolonych w kolejce	
MQIA_MSG_DELIVERY_SEQUENCE	MQLONG	Określa, czy priorytet komunikatu jest istotny	
MQIA_NPM_CLASS	MQLONG	Poziom niezawodności komunikatów nietrwałych	
MQIA_OPEN_INPUT_COUNT	MQLONG	Liczba wywołań MQOPEN , które mają otwartą kolejkę na wejście	
MQIA_OPEN_OUTPUT_COUNT	MQLONG	Liczba wywołań MQOPEN , dla których kolejka była otwarta do wyprowadzania	
MQIA_PROPERTY_CONTROL	MQLONG	Atrybut elementu sterującego właściwości	
MQIA_Q_DEPTH_HIGH_EVENT	MQLONG	Atrybut sterujący dla zdarzeń dużego zapełnienia kolejki	NIEz/OS
MQIA_Q_DEPTH_HIGH_LIMIT	MQLONG	Górny limit głębokości kolejki	NIEz/OS
MQIA_Q_DEPTH_LOW_EVENT	MQLONG	Atrybut sterujący dla zdarzeń niskiego zapełnienia kolejki	NIEz/OS
MQIA_Q_DEPTH_LOW_LIMIT	MQLONG	Dolny limit głębokości kolejki	NIEz/OS

Tabela 549. Selektory atrybutów MQINQ dla kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQIA_Q_DEPTH_MAX_EVENT	MQLONG	Atrybut sterujący dla zdarzeń maksymalnego zapełnienia kolejki	NIEz/OS
MQIA_Q_SERVICE_INTERVAL	MQLONG	Limit czasu obsługi kolejki	NIEz/OS
MQIA_Q_SERVICE_INTERVAL_EVENT	MQLONG	Atrybut sterujący dla zdarzeń odstępu czasu usługi kolejki	NIEz/OS
MQIA_Q_TYPE	MQLONG	Typ kolejki	
MQIA_QSG_DISP	MQLONG	Dyspozycja grupy współużytkowania kolejki	z/OS
MQIA_RETENTION_INTERVAL	MQLONG	Odstęp czasu przechowywania kolejki	
MQIA_SCOPE	MQLONG	Zasięg definicji kolejki	NIEz/OS
MQIA_SHAREABILITY	MQLONG	Określa, czy kolejka może być współużytkowana dla wejścia	
MQIA_STATISTICS_Q	MQLONG	Steruje gromadzeniem danych statystycznych dla kolejki	NIEz/OS
MQIA_TRIGGER_CONTROL	MQLONG	Kontrola wyzwalacza	
MQIA_TRIGGER_DEPTH	MQLONG	Wyzwalacz uruchamiany zapełnieniem	
MQIA_TRIGGER_MSG_PRIORITY	MQLONG	Próg priorytetu komunikatu dla wyzwalacza.	
MQIA_TRIGGER_TYPE	MQLONG	Typ wyzwalacza	
MQIA_USAGE	MQLONG	Użycie	

Tabela 550. Selektory atrybutów MQINQ dla list nazw

Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Czas ostatniej zmiany	
MQCA_NAMELIST_DESC	MQ_NAMELIST_DESC_LENGTH	Opis listy nazw	
MQCA_NAMELIST_NAME	MQ_NAMELIST_NAME_LENGTH	Nazwa obiektu listy nazw	
MQIA_NAMELIST_TYPE	MQLONG	Typ listy nazw	z/OS
MQCA_NAMES	MQ_Q_NAME_LENGTH x *Number of names in the list*	Nazwy na liście nazw	
MQIA_NAME_COUNT	MQLONG	Liczba nazw na liście nazw	
MQIA_QSG_DISP	MQLONG	Dyspozycja grupy współużytkowania kolejki	z/OS

Tabela 551. Selektory atrybutów MQINQ dla definicji procesów

Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Czas ostatniej zmiany	
MQCA_APPL_ID	MQ_PROCESS_APPL_ID_LENGTH	Identyfikator aplikacji	
MQCA_ENV_DATA	MQ_PROCESS_ENV_DATA_LENGTH	Dane środowiska	
MQCA_PROCESS_DESC	MQ_PROCESS_DESC_LENGTH	Opis definicji procesu	
MQCA_PROCESS_NAME	MQ_PROCESS_NAME_LENGTH	Nazwa definicji procesu	
MQCA_USER_DATA	MQ_PROCESS_USER_DATA_LENGTH	Dane użytkownika	
MQIA_APPL_TYPE	MQLONG	Typ aplikacji	
MQIA_QSG_DISP	MQLONG	Dyspozycja grupy współużytkowania kolejki	z/OS

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek

Selektor	Długość pola	Opis	Uwaga
MQCA_ALTERATION_DATE	MQ_DATE_LENGTH	Data ostatniej zmiany	
MQCA_ALTERATION_TIME	MQ_TIME_LENGTH	Czas ostatniej zmiany	
MQCA_CHANNEL_AUTO_DEF_EXIT	MQ_EXIT_NAME_LENGTH	Nazwa wyjścia definicji kanału automatycznego	
MQCA_CHINIT_SERVICE_PARM		Zarezerwowane do użytku przez IBM	
MQCA_CLUSTER_WORKLOAD_DATA	MQ_EXIT_DATA_LENGTH	Dane przekazywane do wyjścia obciążenia klastra	
MQCA_CLUSTER_WORKLOAD_EXIT	MQ_EXIT_NAME_LENGTH	Nazwa wyjścia obciążenia klastra	
MQCA_COMMAND_INPUT_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki wejściowej komendy systemowej	
MQCA_CUSTOM	MQ_CUSTOM_LENGTH	Atrybut niestandardowy dla nowych funkcji	
MQCA_DEAD_LETTER_Q_NAME	MQ_Q_NAME_LENGTH	Nazwa kolejki niedostarczonych komunikatów	
MQCA_DEF_XMIT_Q_NAME	MQ_Q_NAME_LENGTH	Domyślna nazwa kolejki transmisji	

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQCA_DNS_GROUP	MQ_DNS_GROUP_NAME_LENGTH	Nazwa grupy dla programu nasłuchującego TCP, który obsługuje transmisje przychodzące dla grupy współużytkowania kolejek, która ma zostać dołączona. Nazwa ma zastosowanie w przypadku korzystania z usług Dynamic Domain Name Services programu Workload Manager.	z/OS
MQCA_IGQ_USER_ID	MQ_USER_ID_LENGTH	Identyfikator użytkownika kolejkowania wewnątrz grupy	z/OS
V 9.3.0 V 9.3.0 MQCA_INITIAL_KEY	MQ_INITIAL_KEY_LENGTH	Klucz początkowy dla systemu zabezpieczania hasłem	Zwraca wartość ********** , jeśli nie jest pusta, lub wartość pustą, jeśli używany jest domyślny klucz początkowy.
MQCA_INSTALLATION_DESC	MQ_INSTALLATION_DESC_LENGTH	Opis powiązanej instalacji	Nie z/OS. NieIBM i
MQCA_INSTALLATION_NAME	MQ_INSTALLATION_NAME_LENGTH	Nazwa instalacji powiązanej z menedżerem kolejek	Nie z/OS. NieIBM i

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQCA_INSTALLATION_PATH	MQ_INSTALLATION_PATH_LENGTH	Ścieżka, w której jest zainstalowany powiązany produkt IBM MQ	Nie z/OS. NieIBM i
MQCA_LU_GROUP_NAME	MQ_LU_NAME_LENGTH	Ogólna nazwa jednostki logicznej nasłuchiwania jednostki logicznej 6.2 obsługującej transmisje przychodzące dla grupy współużytkowania kolejek, która ma być używana	z/OS
MQCA_LU_NAME	MQ_LU_NAME_LENGTH	Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 . Ustaw tę nazwę na tę samą jednostkę logiczną, która jest używana przez program nasłuchujący do transmisji przychodzących	z/OS
MQCA_LU62_ARM_SUFFIX	MQ_ARM_SUFFIX_LENGTH	Przyrostek SYS1.PARMLIB elementu APPCPM *xx*, który wyznacza LUADD dla tego inicjatora kanału	z/OS
MQCA_PARENT	MQ_Q_MGR_NAME_LENGTH	Nazwa hierarchicznie połączonego menedżera kolejek, który jest nominowany jako nadrzędny dla tego menedżera kolejek	
MQCA_Q_MGR_DESC	MQ_Q_MGR_DESC_LENGTH	Opis menedżera kolejek	
MQCA_Q_MGR_IDENTIFIER	MQ_Q_MGR_IDENTIFIER_LENGT	Identyfikator menedżera kolejek (H)	
MQCA_Q_MGR_NAME	MQ_Q_MGR_NAME_LENGTH	Nazwa lokalnego menedżera kolejek	
MQCA_QSG_NAME	MQ_QSG_NAME_LENGTH	Nazwa grupy współużytkowania kolejek	z/OS
MQCA_REPOSITORY_NAME	MQ_CLUSTER_NAME_LENGTH	Nazwa klastra, dla którego menedżer kolejek udostępnia usługi repozytorium	
MQCA_REPOSITORY_NAMELIST	MQ_NAMELIST_NAME_LENGTH	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których menedżer kolejek udostępnia usługi repozytorium	

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
V 9.3.0 V 9.3.0 MQCA_SSL_KEY_REPO_PASSWORD	MQ_SSL_ENCRYP_KEY_REPO_PWD_LEN	Hasło repozytorium kluczy	Zwraca wartość ********, jeśli nie jest pusta, lub wartość pustą, jeśli nie jest ustawiona. Szyfrowany, gdy jest ustawiony przed zapisaniem.
MQCA_TCP_NAME	MQ_TCP_NAME_LENGTH	Nazwa używanego systemu TCP/IP	z/OS
MQIA_ACCOUNTING_CONN_OVERRIDE	MQLONG	Nadpisz ustawienia rozliczania	NIEz/OS
MQIA_ACCOUNTING_INTERVAL	MQLONG	Częstotliwość zapisywania pośrednich rekordów rozliczeniowych	NIEz/OS
MQIA_ACCOUNTING_MQI	MQLONG	Steruje gromadzeniem informacji rozliczeniowych dla danych MQI	NIEz/OS
MQIA_ACCOUNTING_Q	MQLONG	Steruje kolekcjonowaniem informacji rozliczeniowych dla kolejek	NIEz/OS
MQIA_ACTIVE_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie	z/OS

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)			
Selektor	**Długość pola**	**Opis**	**Uwaga**
MQIA_ADOPTNEWMCA_CHECK	MQLONG	Elementy, które są sprawdzane w celu określenia, czy należy adoptować agent MCA. Sprawdzenie jest wykonywane po wykryciu nowego kanału przychodzącego, który ma taką samą nazwę jak agent MCA, który jest już aktywny.	z/OS
MQIA_ADOPTNEWMCA_INTERVAL	MQLONG	Czas (w sekundach), przez który nowy kanał oczekuje na zakończenie osieroconego kanału	NIEz/OS
MQIA_ADOPTNEWMCA_TYPE	MQLONG	Określa, czy automatycznie restartować osieroconą instancję agenta MCA określonego typu kanału w przypadku wykrycia nowego żądania kanału przychodzącego zgodnego z parametrami AdoptNewMCACheck .	z/OS
MQIA_AUTHORITY_EVENT	MQLONG	Atrybut sterujący dla zdarzeń uprawnień	NIEz/OS
MQIA_BRIDGE_EVENT	MQLONG	Atrybut sterujący dla zdarzeń mostu IMS	z/OS
MQIA_CHANNEL_AUTO_DEF	MQLONG	Atrybut sterujący dla automatycznej definicji kanału	NIEz/OS
MQIA_CHANNEL_AUTO_DEF_EVENT	MQLONG	Atrybut sterujący dla zdarzeń automatycznej definicji kanału	NIEz/OS
MQIA_CHANNEL_EVENT	MQLONG	Atrybut sterujący dla zdarzeń kanału	
MQIA_CHINIT_ADAPTERS	MQLONG	Liczba podzadań adaptera, które mają być używane do przetwarzania wywołań IBM MQ	z/OS
MQIA_CHINIT_DISPATCHERS	MQLONG	Liczba programów rozsyłających, które mają być używane dla inicjatora kanału	z/OS
MQIA_CHINIT_TRACE_AUTO_START	MQLONG	Automatyczne uruchamianie śledzenia inicjatora kanału	z/OS
MQIA_CHINIT_TRACE_TABLE_SIZE	MQLONG	Wielkość obszaru danych śledzenia (w MB) inicjatora kanału	z/OS
MQIA_CLUSTER_WORKLOAD_LENGTH	MQLONG	Długość obciążenia klastra.	
MQIA_CLWL_MRU_CHANNELS	MQLONG	Liczba ostatnio używanych kanałów na potrzeby równoważenia obciążenia klastra	
MQIA_CLWL_USEQ	MQLONG	Użyj kolejek zdalnych	
MQIA_CODED_CHAR_SET_ID	MQLONG	Identyfikator kodowanego zestawu znaków	

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQIA_COMMAND_EVENT	MQLONG	Atrybut sterujący dla zdarzeń komendy	
MQIA_COMMAND_LEVEL	MQLONG	Poziom komend obsługiwany przez menedżer kolejek	
MQIA_CONFIGURATION_EVENT	MQLONG	Atrybut sterujący dla zdarzeń konfiguracji	NIEz/OS
MQIA_DEF_CLUSTER_XMIT_Q_TYPE	MQLONG	Domyślny typ kolejki transmisji, która ma być używana w przypadku kanałów nadawczych klastra.	
MQIA_DIST_LISTS	MQLONG	Obsługa listy dystrybucyjnej	NIEz/OS
MQIA_DNS_WLM	MQLONG	Określa, czy program nasłuchujący TCP, który obsługuje transmisje przychodzące dla grupy współużytkowania kolejek, jest rejestrowany w programie Workload Manager for Dynamic Domain Name Services.	z/OS
MQIA_EXPIRY_INTERVAL	MQLONG	Odstęp czasu między operacjami skanowania komunikatów, które utraciły ważność	z/OS
MQIA_GROUP_UR	MQLONG	Atrybut sterujący określający, czy dla tego menedżera kolejek włączono jednostki odtwarzania GROUP. Dyspozycja jednostki odtwarzania GROUP jest dostępna tylko wtedy, gdy menedżer kolejek jest elementem grupy współużytkowania kolejek.	z/OS
MQIA_IGQ_PUT_AUTHORITY	MQLONG	Uprawnienie do umieszczania w kolejce wewnątrz grupy	z/OS
MQIA_INHIBIT_EVENT	MQLONG	Atrybut sterujący dla zdarzeń blokowania	NIEz/OS
MQIA_INTRA_GROUP_queuing	MQLONG	Obsługa kolejkowania wewnątrz grupy	z/OS
MQIA_LISTENER_TIMER	MQLONG	Odstęp czasu (w sekundach) między kolejnymi próbami zrestartowania procesu nasłuchiwania podejmowanymi przez IBM MQ w przypadku niepowodzenia komunikacji APPC lub TCP/IP.	z/OS
MQIA_LOCAL_EVENT	MQLONG	Atrybut sterujący dla zdarzeń lokalnych	NIEz/OS
MQIA_LOGGER_EVENT	MQLONG	Atrybut sterujący dla zdarzeń blokowania	NIEz/OS

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQIA_LU62_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być bieżące, lub klientów, które mogą być połączone, za pomocą protokołu transmisji LU 6.2	z/OS
MQIA_MSG_MARK_BROWSE_INTERVAL	MQLONG	Odstęp czasu (w milisekundach), po upływie którego menedżer kolejek może automatycznie usunąć znacznik z komunikatów przeglądania. **Ostrzeżenie:** Nie należy ustawiać tej wartości poniżej wartości domyślnej 5000.	
MQIA_MAX_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być bieżące (w tym kanały połączenia z serwerem z połączonymi klientami)	z/OS
MQIA_MAX_HANDLES	MQLONG	Maksymalna liczba uchwytów	
MQIA_MAX_MSG_LENGTH	MQLONG	Maksymalna długość komunikatu	
MQIA_MAX_PRIORITY	MQLONG	Maksymalny priorytet	
MQIA_MAX_UNCOMMITTED_MSGS	MQLONG	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy	
MQIA_OUTBOUND_PORT_MAX	MQLONG	W przypadku parametru MQIA_OUTBOUND_PORT_MINdefiniuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.	z/OS
MQIA_OUTBOUND_PORT_MIN	MQLONG	W przypadku parametru MQIA_OUTBOUND_PORT_MAXdefiniuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.	z/OS
MQIA_PERFORMANCE_EVENT	MQLONG	Atrybut sterujący dla zdarzeń wydajności	NIEz/OS
MQIA_PLATFORM	MQLONG	Platforma, na której rezyduje menedżer kolejek	
MQIA_PROT_POLICY_CAPABILITY	MQLONG	Wskazuje, czy możliwości zabezpieczeń produktu Advanced Message Security są dostępne dla menedżera kolejek.	
MQIA_PUBSUB_MAXMSG_RETRY_COUNT	MQLONG	Liczba prób ponownego przetworzenia komunikatu komendy zakończonej niepowodzeniem w punkcie synchronizacji	

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)

Selektor	Długość pola	Opis	Uwaga
MQIA_PUBSUB_MODE	MQLONG	Określa, czy mechanizm publikowania/subskrypcji i umieszczony w kolejce interfejs publikowania/subskrypcji są uruchomione. Aplikacje do publikowania lub subskrybowania przy użyciu aplikacyjnego interfejsu programistycznego wymagają mechanizmu publikowania/subskrybowania. Kolejki monitorowane przez umieszczony w kolejce interfejs publikowania/subskrypcji wymagają uruchomienia umieszczonego w kolejce interfejsu publikowania/subskrypcji.	
MQIA_PUBSUB_NP_MSG	MQLONG	Czy odrzucić (lub zachować) niedostarczony komunikat wejściowy	
MQIA_PUBSUB_NP_RESP	MQLONG	Steruje zachowaniem niedostarczonych komunikatów odpowiedzi	
MQIA_PUBSUB_SYNC_PT	MQLONG	Określa, czy tylko trwałe (lub wszystkie) komunikaty są przetwarzane w punkcie synchronizacji	
MQIA_QMGR_CFCONLOS	MQLONG	Określa działanie, które ma zostać wykonane, gdy menedżer kolejek utraci połączenie ze strukturą administracyjną lub dowolnymi strukturami narzędzia CF z wartością CFCONLOS ustawioną na ASQMGR .	z/OS
MQIA_RECEIVE_TIMEOUT	MQLONG	Przybliżony czas oczekiwania kanału TCP/IP na odebranie danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego. Jest to wartość liczbowa, kwalifikowana przez MQIA_RECEIVE_TIMEOUT_TYPE.	z/OS
MQIA_RECEIVE_TIMEOUT_MIN	MQLONG	Minimalny czas oczekiwania kanału TCP/IP na odebranie danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywności	z/OS
MQIA_RECEIVE_TIMEOUT_TYPE	MQLONG	Przybliżony czas oczekiwania kanału TCP/IP na odebranie danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego. MQIA_RECEIVE_TIMEOUT_TYPE jest kwalifikatorem zastosowanym do MQIA_RECEIVE_TIMEOUT.	z/OS

Tabela 552. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)			
Selektor	**Długość pola**	**Opis**	**Uwaga**
MQIA_REMOTE_EVENT	MQLONG	Atrybut sterujący dla zdarzeń zdalnych	NIEz/OS
MQIA_SECURITY_CASE	MQLONG	Przypadek profili zabezpieczeń	z/OS
MQIA_SSL_EVENT	MQLONG	Atrybut sterujący dla zdarzeń kanału	
MQIA_SSL_FIPS_REQUIRED	MQLONG	Do szyfrowania używaj tylko algorytmów z certyfikatem FIPS	
MQIA_SSL_RESET_COUNT	MQLONG	Licznik resetowania klucza TLS	
MQIA_START_STOP_EVENT	MQLONG	Atrybut sterujący dla zdarzeń uruchomienia i zatrzymania	NIEz/OS
MQIA_STATISTICS_AUTO_CLUSSDR	MQLONG	Steruje gromadzeniem informacji o monitorowaniu statystyk dla kanałów nadawczych klastra	
MQIA_STATISTICS_CHANNEL	MQLONG	Steruje gromadzeniem danych statystycznych dla kanałów	
MQIA_STATISTICS_INTERVAL	MQLONG	Jak często mają być zapisywane dane monitorowania statystyk	NIEz/OS
MQIA_STATISTICS_MQI	MQLONG	Steruje gromadzeniem informacji dotyczących monitorowania statystyk dla menedżera kolejek	NIEz/OS
MQIA_STATISTICS_Q	MQLONG	Steruje gromadzeniem danych statystycznych dla kolejek	NIEz/OS
MQIA_SYNCPOINT	MQLONG	dostępność punktu synchronizacji	
MQIA_TCP_CHANNELS	MQLONG	Maksymalna liczba kanałów, które mogą być bieżące, lub klientów, które mogą być połączone, przy użyciu protokołu transmisji TCP/IP	z/OS
MQIA_TCP_KEEP_ALIVE	MQLONG	Czy użyć narzędzia TCP KEEPALIVE do sprawdzenia, czy drugi koniec połączenia jest nadal dostępny	z/OS
MQIA_TCP_STACK_TYPE	MQLONG	Określa, czy inicjator kanału może używać tylko przestrzeni adresowej TCP/IP określonej w parametrze TCPNAME, czy też opcjonalnie może być powiązany z dowolnym wybranym adresem TCP/IP.	z/OS
MQIA_TRACE_ROUTE_RECORDING	MQLONG	Steruje rejestrowaniem informacji o trasie śledzenia	z/OS
MQIA_TREE_LIFE_TIME	MQLONG	Czas życia nieużywanych tematów nieadministracyjnych	
MQIA_TRIGGER_INTERVAL	MQLONG	Interwał wyzwalacza	

IntAttrLiczba
Typ: MQLONG -wejście

Jest to liczba elementów w tablicy *IntAttrs* . Zero jest poprawną wartością.

Jeśli wartość IntAttrCount jest co najmniej liczbą selektorów MQIA_* w parametrze **Selectors** , zwracane są wszystkie żądane atrybuty całkowitoliczbowe.

IntAttrs
Typ: MQLONG x *IntAttrCount* -dane wyjściowe

Jest to tablica wartości atrybutów całkowitoliczbowych *IntAttrCount* .

Wartości atrybutów całkowitych są zwracane w tej samej kolejności, co selektory MQIA_* w parametrze **Selectors** . Jeśli tablica zawiera więcej elementów niż liczba selektorów MQIA_* , nadmiarowe elementy pozostają niezmienione.

Jeśli *Hobj* reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, zwracana jest konkretna wartość MQIAV_NOT_APPLICABLE . Jest on zwracany dla odpowiedniego elementu w tablicy *IntAttrs* .

Jeśli parametr **IntAttrCount** lub **SelectorCount** ma wartość zero, nie jest przywoływany parametr *IntAttrs* . W takim przypadku adres parametru przekazany przez programy napisane w języku C lub S/390 może mieć wartość NULL.

CharAttrDługość
Typ: MQLONG -wejście

Jest to długość parametru **CharAttrs** w bajtach.

CharAttrDługość musi być co najmniej sumą długości żądanych atrybutów znakowych (patrz sekcja Selektory). Zero jest poprawną wartością.

CharAttrs
Typ: MQCHAR x *CharAttrLength* -dane wyjściowe

Jest to bufor, w którym zwracane są atrybuty znakowe, połączone ze sobą. Długość buforu jest określona przez parametr **CharAttrLength** .

Atrybuty znakowe są zwracane w tej samej kolejności, co selektory MQCA_* w parametrze **Selectors** . Długość każdego łańcucha atrybutu jest ustalona dla każdego atrybutu (patrz sekcja Selektory), a jego wartość jest w razie potrzeby dopełniana do prawej strony odstępami. Można podać bufor większy niż wymagany, aby zawierał wszystkie żądane atrybuty znaków i dopełnianie. Bajty wykraczające poza ostatnią zwróconą wartość atrybutu nie zostały zmienione.

Jeśli *Hobj* reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, zwracany jest łańcuch znaków składający się wyłącznie z gwiazdek (*). Gwiazdka jest zwracana jako wartość tego atrybutu w *CharAttrs*.

Jeśli parametr *CharAttrLength* lub **SelectorCount** ma wartość zero, nie jest przywoływany parametr *CharAttrs* . W takim przypadku adres parametru przekazany przez programy napisane w języku C lub S/390 może mieć wartość NULL.

CompCode
Typ: MQLONG -dane wyjściowe

Kod zakończenia:

MQCC_OK
Zakończono pomyślnie.

MQCC_WARNING
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_NONE
(0, X'000') Nie ma powodu do zgłaszania.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
(2008, X'7D8') Niewystarczająca ilość miejsca dozwolona dla atrybutów znakowych.

MQRC_INT_ATTR_COUNT_TOO_SMALL
(2022, X'7E6') Niewystarczająca ilość miejsca dozwolona dla atrybutów całkowitoliczbowych.

MQRC_SELECTOR_NOT_FOR_TYPE
(2068, X'814') Selektor nie ma zastosowania do typu kolejki.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NOT_AVAILABLE
(2204, X'89C') Adapter jest niedostępny.

MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852') Nie można załadować modułu usługi adaptera.

MQRC_API_EXIT_ERROR
(2374, X'946') Wyjście funkcji API nie powiodło się.

MQRC_API_EXIT_LOAD_ERROR
(2183, X'887') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne identyfikatory ASID różnią się.

MQRC_CALL_IN_PROGRESS
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_STRUC_FAILED
(2373, X'945') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_USE
(2346, X'92A') Struktura narzędzia CF w użyciu.

MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Niepoprawna długość atrybutów znakowych.

MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Niepoprawny łańcuch atrybutów znakowych.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_STOPPING
(2203, X'89B') Połączenie jest zamykane.

MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Niepoprawna liczba atrybutów całkowitych.

MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Niepoprawna tablica atrybutów całkowitych.

MQRC_NOT_OPEN_FOR_INQUIRE
(2038, X'7F6') Kolejka nie jest otwarta do odpytywania.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_DAMAGED
(2101, X'835') Obiekt uszkodzony.

MQRC_PAGESET_ERROR
(2193, X'891') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED
(2052, X'804') Kolejka została usunięta.

MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_STOPPING
(2162, X'872') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836') Niewystarczające zasoby systemowe.

MQRC_SELECTOR_COUNT_ERROR
(2065, X'811') Niepoprawna liczba selektorów.

MQRC_SELECTOR_ERROR
(2067, X'813') Selektor atrybutów jest niepoprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812') Zbyt duża liczba selektorów.

MQRC_STORAGE_NOT_AVAILABLE
(2071, X'817') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie wstrzymane przez program obsługi wyjścia.

MQRC_UNEXPECTED_ERROR
(2195, X'893') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn .

Użycie notatek

1. Zwracane wartości są obrazem stanu wybranych atrybutów. Nie ma gwarancji, że atrybuty pozostaną takie same, zanim aplikacja będzie mogła działać na zwróconych wartościach.
2. Po otwarciu kolejki modelowej tworzona jest dynamiczna kolejka lokalna. Dynamiczna kolejka lokalna jest tworzona nawet po otwarciu kolejki modelowej w celu uzyskania informacji o jej atrybutach.

   Atrybuty kolejki dynamicznej są w dużej mierze takie same jak atrybuty kolejki modelowej w momencie tworzenia kolejki dynamicznej. Jeśli następnie zostanie użyte wywołanie MQINQ dla tej kolejki, menedżer kolejek zwróci atrybuty kolejki dynamicznej, a nie atrybuty kolejki modelowej. Szczegółowe informacje o tym, które atrybuty kolejki modelowej są dziedziczone przez kolejkę dynamiczną, zawiera sekcja Tabela 561 na stronie 865 .
3. Jeśli obiekt, którego dotyczy zapytanie, jest kolejką aliasową, wartości atrybutów zwracane przez wywołanie MQINQ są atrybutami kolejki aliasowej. Nie są to atrybuty podstawowej kolejki lub tematu, na które jest tłumaczony alias.
4. Jeśli obiekt, którego dotyczy zapytanie, jest kolejką klastra, atrybuty, które można sprawdzić, zależą od sposobu otwarcia kolejki:

   - Można otworzyć kolejkę klastra w celu wykonania zapytania oraz jedną lub więcej operacji wejściowych, przeglądania lub ustawiania. W tym celu musi istnieć lokalna instancja kolejki klastra, aby otwarcie powiodło się. W takim przypadku atrybuty, do których można uzyskać dostęp, są atrybutami poprawnymi dla kolejek lokalnych.

Jeśli kolejka klastra jest otwarta do odpytywania bez określonego wejścia, przeglądania lub ustawienia, wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068), jeśli zostanie podjęta próba sprawdzenia atrybutów, które są poprawne tylko dla kolejek lokalnych, a nie dla kolejek klastra.

- Kolejkę klastra można otworzyć w celu wykonania zapytania podczas przekazywania nazwy podstawowego menedżera kolejek połączonego menedżera kolejek.

  W tym celu musi istnieć lokalna instancja kolejki klastra, aby otwarcie powiodło się. Jeśli podstawowy menedżer kolejek nie zostanie przekazany, wywołanie zwróci kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068), jeśli zostanie podjęta próba sprawdzenia atrybutów, które są poprawne tylko dla kolejek lokalnych, a nie kolejek klastra.

- Jeśli kolejka klastra jest otwarta tylko na potrzeby wykonywania zapytań lub na potrzeby wykonywania zapytań i danych wyjściowych, mogą być wysyłane zapytania tylko do wymienionych atrybutów. Atrybut **QType** ma w tym przypadku wartość MQQT_CLUSTER :

  - MQCA_Q_DESC
  - MQCA_Q_NAME
  - MQIA_DEF_BIND
  - MQIA_DEF_PERSISTENCE
  - MQIA_DEF_PRIORITY
  - MQIA_INHIBIT_PUT
  - MQIA_Q_TYPE

  Kolejkę klastra można otworzyć bez stałego powiązania. Można go otworzyć za pomocą parametru MQOO_BIND_NOT_FIXED określonego w wywołaniu MQOPEN . Alternatywnie można podać parametr MQOO_BIND_AS_Q_DEFi ustawić atrybut **DefBind** kolejki na wartość MQBND_BIND_NOT_FIXED. Jeśli zostanie otwarta kolejka klastra bez stałego powiązania, kolejne wywołania funkcji MQINQ dla tej kolejki mogą powodować zapytania o różne instancje kolejki klastra. Jednak jest to typowe dla wszystkich instancji, które mają takie same wartości atrybutów.

- Dla klastra można zdefiniować obiekt kolejki aliasowej. Ponieważ atrybuty TARGTYPE i TARGET nie są atrybutami klastra, proces wykonujący proces MQOPEN w kolejce aliasowej nie wie o obiekcie, na który alias jest tłumaczony.

  Podczas początkowego wykonywania komendy MQOPENkolejka aliasowa jest tłumaczona na menedżer kolejek i kolejkę w klastrze. Tłumaczenie nazw odbywa się ponownie w zdalnym menedżerze kolejek i w tym miejscu jest tłumaczone pole TARGTPYE kolejki aliasowej.

  Jeśli kolejka aliasowa jest tłumaczona na alias tematu, publikowanie komunikatów umieszczonych w kolejce aliasowej odbywa się w tym zdalnym menedżerze kolejek.

Patrz sekcja Kolejki klastrów .

5. Użytkownik może chcieć uzyskać informacje o wielu atrybutach, a następnie ustawić niektóre z nich przy użyciu wywołania MQSET . Aby program sprawnie sprawdzać i ustawiać atrybuty, które mają być ustawiane na początku tablic selektora. W takim przypadku dla systemu MQSETmożna użyć tych samych macierzy ze zmniejszoną liczbą wystąpień.
6. Jeśli wystąpi więcej niż jedna sytuacja ostrzegawcza (patrz parametr **CompCode**), zwrócony kod przyczyny jest pierwszym z poniższej listy, który ma zastosowanie:
   a. MQRC_SELECTOR_NOT_FOR_TYPE
   b. MQRC_INT_ATTR_COUNT_TOO_SMALL
   c. MQRC_CHAR_ATTRS_TOO_SHORT
7. Informacje na temat atrybutów obiektu znajdują się w następującym temacie:
   - “Atrybuty kolejek” na stronie 863
   - “Atrybuty listy nazw” na stronie 898
   - “Atrybuty definicji procesów” na stronie 900

- “Atrybuty menedżera kolejek” na stronie 824

Wywołanie C

```
MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
     CALL 'MQINQ' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                        INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                        CHARATTRS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQINQ (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
        CALL MQINQ,(HCONN,HOBJ,SELECTORCOUNT,SELECTORS,INTATTRCOUNT,  X
              INTATTRS,CHARATTRLENGTH,CHARATTRS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

```
MQINQ Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-właściwość komunikatu zapytania

Wywołanie MQINQMP zwraca wartość właściwości komunikatu.

Składnia

MQINQMP (*Hconn*, *Hmsg*, *InqPropOpc*, *Nazwa*, *PropDesc*, *Typ*, *ValueLength*, *Wartość*, *DataLength*, *CompCode*, *Przyczyna*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony za pomocą wywołania MQHC_UNASSOCIATED_HCONN, należy nawiązać poprawne połączenie w wątku, sprawdzając właściwość uchwytu

komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zerwana wartość MQRC_CONNECTION_BROKEN.

Komunikat Hmsg
Typ: MQHMSG-input

Jest to uchwyt komunikatu, którego dotyczy zapytanie. Wartość została zwrócona przez poprzednie wywołanie funkcji **MQCRTMH** .

InqPropOpcje
Typ: MQIMPO-wejście/wyjście

Szczegółowe informacje można znaleźć w opisie typu danych MQIMPO .

Nazwa
Typ: MQCHARV-wejście/wyjście

Nazwa właściwości, do której ma zostać wykonane zapytanie.

Jeśli nie można znaleźć właściwości o tej nazwie, wywołanie nie powiedzie się z powodu błędu MQRC_PROPERTY_NOT_AVAILABLE.

Na końcu nazwy właściwości można użyć znaku wieloznacznego procentu (%). Znak wieloznaczny zastępuje zero lub więcej znaków, łącznie ze znakiem kropki (.). Dzięki temu aplikacja może uzyskać informacje o wartości wielu właściwości. Wywołaj komendę MQINQMP z opcją MQIMPO_INQ_FIRST, aby uzyskać pierwszą zgodną właściwość, a następnie ponownie z opcją MQIMPO_INQ_NEXT, aby uzyskać następną zgodną właściwość. Jeśli nie są dostępne więcej zgodnych właściwości, wywołanie kończy się niepowodzeniem z błędem MQRC_PROPERTY_NOT_AVAILABLE. Jeśli pole *ReturnedName* struktury InqPropOpts zostało zainicjowane z adresem lub przesunięciem dla zwróconej nazwy właściwości, jest to wykonywane po zwróceniu przez MQINQMP nazwy właściwości, która została dopasowana. Jeśli pole *VSBufSize* w pliku *ReturnedName* w strukturze InqPropOpts jest krótsze niż długość zwróconej nazwy właściwości, kod zakończenia jest ustawiany jako MQCC_FAILED z przyczyną MQRC_PROPERTY_NAME_TOO_BIG.

Właściwości, które mają znane synonimy, są zwracane w następujący sposób:

1. Właściwości z przedrostkiem mqps. są zwracane jako nazwa właściwości IBM MQ . Na przykład "MQTopicString" jest zwróconą nazwą, a nie "mqps.Top"
2. Właściwości z przedrostkiem jms. lub "mcd." są zwracane jako nazwa pola nagłówka JMS , na przykład "JMSExpiration" jest zwróconą nazwą, a nie "jms.Exp".
3. Właściwości z przedrostkiem usr. są zwracane bez tego przedrostka, na przykład zamiast "usr.Color" jest zwracana wartość "Color".

Właściwości z synonimami są zwracane tylko raz.

W języku programowania C następujące zmienne makra są zdefiniowane w celu uzyskiwania informacji o wszystkich właściwościach, a następnie o wszystkich właściwościach, które rozpoczynają się od "usr.":

MQPROP_INQUIRE_ALL
Sprawdź wszystkie właściwości komunikatu.

Właściwości MQPROP_INQUIRE_ALL można użyć w następujący sposób:

```
MQCHARV Name = {MQPROP_INQUIRE_ALL};
```

MQPROP_INQUIRE_ALL_USR
Sprawdź wszystkie właściwości komunikatu rozpoczynające się od łańcucha usr. Zwrócona nazwa jest zwracana bez "usr." prefiks.

Jeśli podano parametr MQIMP_INQ_NEXT, ale nazwa uległa zmianie od poprzedniego wywołania lub jest to pierwsze wywołanie, to domyślnie przyjmowany jest parametr MQIMPO_INQ_FIRST.

Więcej informacji na temat używania nazw właściwości zawiera sekcja Nazwy właściwości i sekcja Ograniczenia dotyczące nazw właściwości .

PropDesc
Typ: MQPD-wyjście

Ta struktura jest używana do definiowania atrybutów właściwości, w tym informacji o tym, co się stanie, jeśli właściwość nie jest obsługiwana, kontekstu komunikatu, do którego należy właściwość oraz komunikatów, do których właściwość powinna zostać skopiowana. Szczegółowe informacje na temat tej struktury zawiera sekcja MQPD .

Typ
Typ: MQLONG-input/output

Po powrocie z wywołania MQINQMP ten parametr jest ustawiany na typ danych *Wartość*. Typ danych może być dowolny z następujących typów:

MQTYPE_BOOLEAN (wartość boolowska)
Wartość boolowska.

MQTYPE_BYTE_STRING ŁAŃCUCH
Łańcuch bajtowy.

MQTYPE_INT8
8-bitowa liczba całkowita ze znakiem.

MQTYPE_INT16
16-bitowa liczba całkowita ze znakiem.

MQTYPE_INT32
32-bitowa liczba całkowita ze znakiem.

MQTYPE_INT64
64-bitowa liczba całkowita ze znakiem.

MQTYPE_FLOAT32
32-bitowa liczba zmiennopozycyjna.

MQTYPE_FLOAT64
64-bitowa liczba zmiennopozycyjna.

MQTYPE_STRING (łańcuch MQTYPE)
Łańcuch znaków.

MQTYPE_NULL
Właściwość istnieje, ale ma wartość null.

Jeśli typ danych wartości właściwości nie zostanie rozpoznany, zostanie zwrócona wartość MQTYPE_STRING i w obszarze *Wartość* zostanie umieszczona reprezentacja łańcuchowa wartości. Reprezentację łańcuchową typu danych można znaleźć w polu *TypeString* parametru *InqPropOpts* . Zwracany jest kod zakończenia ostrzeżenia z przyczyną MQRC_PROP_TYPE_NOT_SUPPORTED.

Ponadto, jeśli określono opcję MQIMPO_CONVERT_TYPE, żądana jest konwersja wartości właściwości. Użyj opcji *Typ* jako danych wejściowych, aby określić typ danych, który ma być zwracany przez właściwość. Szczegółowe informacje na temat konwersji typów danych zawiera opis opcji MQIMPO_CONVERT_TYPE struktury MQIMPO .

Jeśli konwersja typów nie jest żądana, na wejściu można użyć następującej wartości:

MQTYPE_AS_SET
Wartość właściwości jest zwracana bez przekształcania jej typu danych.

ValueLength
Typ: MQLONG-input

Długość obszaru wartości w bajtach. Należy podać zero dla właściwości, dla których nie jest wymagana wartość zwracana. Mogą to być właściwości zaprojektowane przez aplikację tak, aby miały wartość NULL lub pusty łańcuch. Należy również podać wartość zero, jeśli podano opcję MQIMPO_QUERY_LENGTH . W takim przypadku nie jest zwracana żadna wartość.

Wartość
Typ: MQBYTEx *ValueLength* -dane wyjściowe

Jest to obszar, w którym ma być zawarta wartość właściwości, do której wysłano zapytanie. Bufor powinien być wyrównany do granicy odpowiedniej dla zwracanej wartości. Jeśli ta czynność nie powiedzie się, może to spowodować błąd podczas późniejszego uzyskiwania dostępu do tej wartości.

Jeśli wartość *ValueLength* jest mniejsza niż długość wartości właściwości, to do pola *Wartość* zostanie przeniesiona jak najwięcej wartości właściwości, a wywołanie zakończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i przyczyną MQRC_PROPERTY_VALUE_TOO_BIG.

Zestaw znaków danych w polu *Wartość* jest określony przez pole ReturnedCCSID w parametrze InqPropOpts. Kodowanie danych w polu *Wartość* jest określone w polu ReturnedEncoding w parametrze InqPropOpts.

W języku programowania C parametr jest zadeklarowany jako wskaźnik do unieważnienia; jako parametr można podać adres dowolnego typu danych.

Jeśli parametr *ValueLength* ma wartość zero, *Value* nie jest przywoływany, a jego wartość przekazywana przez programy napisane w języku C lub System/390 może mieć wartość NULL.

DataLength
Typ: MQLONG-wyjście

Jest to długość (w bajtach) rzeczywistej wartości właściwości zwróconej w obszarze *Wartość* .

Jeśli wartość *DataLength* jest mniejsza niż długość wartości właściwości, *DataLength* jest nadal wypełniane po powrocie z wywołania MQINQMP. Dzięki temu aplikacja może określić wielkość buforu wymaganą do dostosowania wartości właściwości, a następnie ponownie wywołać wywołanie z buforem o odpowiedniej wielkości.

Można również zwrócić następujące wartości.

Jeśli parametr *Typ* jest ustawiony na wartość MQTYPE_STRING lub MQTYPE_BYTE_STRING:

MQVL_EMPTY_STRING
Właściwość istnieje, ale nie zawiera znaków ani bajtów.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_PROP_NAME_NOT_CONVERTED (nieskonwertowany)
(2492, X'09BC') Zwrócona nazwa właściwości nie została przekształcona.

MQRC_PROP_VALUE_NOT_CONVERTED
(2466, X'09A2') Wartość właściwości nie została przekształcona.

MQRC_PROP_TYPE_NOT_SUPPORTED
(2467, X'09A3') Typ danych właściwości nie jest obsługiwany.

BŁĄD MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'086D') Główne identyfikatory ASID są różne.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'07D4') Niepoprawny parametr wartości.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Niepoprawny parametr długości wartości.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') Niepoprawny parametr długości danych.

BŁĄD MQRC_IMPO_ERROR
(2464, X'09A0') Niepoprawna struktura opcji właściwości komunikatu.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

MQRC_MSG_HANDLE_IN_USE,
(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07F8') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_PD_ERROR
(2482, X'09B2') Niepoprawna struktura deskryptora właściwości.

MQRC_PROP_CONV_NOT_SUPPORTED
(2470, X'09A6') Konwersja rzeczywistego na żądany typ danych nie jest obsługiwana.

BŁĄD MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Niepoprawna nazwa właściwości.

MQRC_PROPERTY_NAME_TOO_BIG
(2465, X'09A1') Nazwa właściwości jest zbyt duża dla zwróconego buforu nazw.

WŁAŚCIWOŚĆ_MQRC_NIEDOSTĘPNA
(2471, X'09A7) Właściwość niedostępna.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Wartość właściwości jest zbyt duża dla obszaru wartości.

MQRC_PROP_NUMBER_FORMAT_BŁĄD
(2472, X'09A8') Wystąpił błąd formatu liczb w danych wartości.

BŁĄD TYPU WŁAŚCIWOŚCI MQRC_PROPERTY_TYPE_ERROR
(2473, X'09A9') Niepoprawny żądany typ właściwości.

BŁĄD MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Niepoprawny identyfikator kodowanego zestawu znaków nazwy właściwości.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'0871 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Wywołanie C

```
MQINQMP (Hconn, Hmsg, &InqPropOpts, &Name, &PropDesc, &Type,
ValueLength, Value, &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQINQMP' USING HCONN, HMSG, INQMSGOPTS, NAME, PROPDESC, TYPE,
VALUELENGTH, VALUE, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQINQMP (Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type,
ValueLength, Value, DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQINQMP,(HCONN,HMSG,INQMSGOPTS,NAME,PROPDESC,TYPE,
VALUELENGTH,VALUE,DATALENGTH,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-przekształcenie uchwytu komunikatu w bufor

Wywołanie MQMHBUF przekształca uchwyt komunikatu w bufor i jest odwrotnością wywołania MQBUFMH.

Składnia

MQMHBUF (*Hconn*, *Hmsg*, *MsgHBufOpts*, *Name*, *MsgDesc*, *BufferLength*, *Buffer*, *DataLength*, *CompCode*, *Reason*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony przy użyciu wywołania MQHC_UNASSOCIATED_HCONN, w wątku usuwającym uchwyt komunikatu musi zostać nawiązane poprawne połączenie. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie nie powiedzie się i zostanie zwrócony komunikat MQRC_CONNECTION_BROKEN.

Komunikat Hmsg
Typ: MQHMSG-input

Jest to uchwyt komunikatu, dla którego wymagany jest bufor. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

Opcje MsgHBuf
Typ: MQMHBO-input

Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia buforów z uchwytów komunikatów.

Szczegółowe informacje można znaleźć w sekcji “MQMHBO-opcje przesyłania komunikatu do buforu” na stronie 491.

Nazwa
Typ: MQCHARV-input

Nazwa właściwości do umieszczenia w buforze.

Jeśli nie można znaleźć właściwości zgodnej z nazwą, wywołanie kończy się niepowodzeniem z błędem MQRC_PROPERTY_NOT_AVAILABLE.

Można użyć znaku wieloznacznego, aby umieścić więcej niż jedną właściwość w buforze. W tym celu należy użyć znaku wieloznacznego '%' na końcu nazwy właściwości. Ten znak wieloznaczny zastępuje zero lub więcej znaków, w tym znak '.' .

W języku programowania C następujące zmienne makra są zdefiniowane w celu uzyskiwania informacji o wszystkich właściwościach i wszystkich właściwościach, które rozpoczynają się od 'usr':

MQPROP_INQUIRE_ALL
Umieść wszystkie właściwości komunikatu w buforze

MQPROP_INQUIRE_ALL_USR
Umieść wszystkie właściwości komunikatu rozpoczynające się od znaków 'usr.' do buforu.

Więcej informacji na temat używania nazw właściwości zawiera sekcja Nazwy właściwości i sekcja Ograniczenia dotyczące nazw właściwości .

MsgDesc
Typ: MQMD-wejście/wyjście

Struktura *MsgDesc* opisuje zawartość obszaru buforu.

W danych wyjściowych pola *Encoding*, *CodedCharSetId* i *Format* są ustawione w taki sposób, aby poprawnie opisywały kodowanie, identyfikator zestawu znaków i format danych w obszarze buforu zapisywanych przez wywołanie.

Dane w tej strukturze są w zestawie znaków i kodowaniu aplikacji.

BufferLength
Typ: MQLONG-input

BufferLength to długość obszaru buforu w bajtach.

Buforuj
Typ: MQBYTExBuffer-długość-dane wyjściowe

Buffer definiuje obszar, który ma zawierać właściwości komunikatu. Bufor należy wyrównać do granicy 4-bajtowej.

Jeśli wartość *BufferLength* jest mniejsza niż długość wymagana do zapisania właściwości w pliku *Buffer*, działanie MQMHBUF kończy się niepowodzeniem z błędem MQRC_PROPERTY_VALUE_TOO_BIG.

Zawartość buforu może ulec zmianie, nawet jeśli wywołanie nie powiedzie się.

DataLength
Typ: MQLONG-wyjście

DataLength to długość (w bajtach) właściwości zwróconych w buforze. Jeśli wartość wynosi zero, żadne właściwości nie są zgodne z wartością podaną w parametrze *Name* , a wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_PROPERTY_NOT_AVAILABLE.

Jeśli wartość *BufferLength* jest mniejsza niż długość wymagana do zapisania właściwości w buforze, wywołanie MQMHBUF nie powiedzie się i zostanie wyświetlony komunikat MQRC_PROPERTY_VALUE_TOO_BIG, ale wartość ta jest nadal wprowadzana w pliku *DataLength*. Umożliwia to aplikacji określenie wielkości buforu wymaganego do dostosowania właściwości, a następnie ponowne wywołanie z wymaganą wartością *BufferLength*.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

BŁĄD MQRC_MHBO_ERROR
(2501, X'095C') Niepoprawna struktura opcji przesyłania komunikatu do buforu.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'07D4') Niepoprawny parametr buforu.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'07DA') Niepoprawny parametr długości danych.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

MQRC_MD_BŁĄD
(2026, X'07EA') Niepoprawny deskryptor komunikatu.

MQRC_MSG_HANDLE_IN_USE,
(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Nazwa właściwości jest niepoprawna.

WŁAŚCIWOŚĆ_MQRC_NIEDOSTĘPNA
(2471, X'09A7') Właściwość niedostępna.

MQRC_PROPERTY_VALUE_TOO_BIG
(2469, X'09A5') Wartość BufferLength jest zbyt mała, aby pomieścić określone właściwości.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Wywołanie C

```
MQMHBUF (Hconn, Hmsg, &MsgHBufOpts, &Name, &MsgDesc, BufferLength, Buffer,
         &DataLength, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Użycie notatek

MQMHBUF przekształca uchwyt komunikatu w bufor.

Można go używać razem z wyjściem funkcji API MQGET w celu uzyskania dostępu do pewnych właściwości przy użyciu funkcji API właściwości komunikatu, a następnie przekazać je w buforze z powrotem do aplikacji zaprojektowanej do używania nagłówków MQRFH2 zamiast uchwytów komunikatów.

To wywołanie jest odwrotnością wywołania MQBUFMH, którego można użyć do przeanalizowania właściwości komunikatu z buforu do uchwytu komunikatu.

Wywołanie COBOL

```
     CALL 'MQMHBUF' USING HCONN, HMSG, MSGHBUFOPTS, NAME, MSGDESC,
                          BUFFERLENGTH, BUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQMHBUF (Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer,
DataLength, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQMHBUF,(HCONN,HMSG,MSGHBUFOPTS,NAME,MSGDESC,BUFFERLENGTH,
               BUFFER,DATALENGTH,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-otwarcie obiektu

Wywołanie MQOPEN zapewnia dostęp do obiektu.

Poprawne są następujące typy obiektów:

- Kolejka (łącznie z listami dystrybucyjnymi)
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- Temat

Składnia

MQOPEN (*Hconn*, *ObjDesc*, *Opcje*, *Hobj*, *CompCode*, *Przyczyna*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość Hconn została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

z/OS W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

ObjDesc
Typ: MQOD-wejście/wyjście

Jest to struktura identyfikująca obiekt, który ma zostać otwarty. Szczegółowe informacje na ten temat zawiera sekcja "MQOD-deskryptor obiektu" na stronie 493 .

Jeśli pole `ObjectName` w parametrze **ObjDesc** jest nazwą kolejki modelowej, jest to dynamiczna kolejka lokalna. jest tworzony z atrybutami kolejki modelowej; dzieje się tak niezależnie od opcji określonych w parametrze **Options** . Kolejne operacje używające `Hobj` zwracane przez wywołanie MQOPEN są wykonywane w nowej kolejce dynamicznej, a nie w kolejce modelowej. Dotyczy to również wywołań MQINQ i MQSET. Nazwa kolejki modelowej w parametrze **ObjDesc** jest zastępowana nazwą utworzonej kolejki dynamicznej. Typ kolejki dynamicznej jest określany na podstawie wartości atrybutu **DefinitionType** kolejki modelowej (patrz "Atrybuty kolejek" na stronie 863). Informacje na temat opcji zamykania mających zastosowanie do kolejek dynamicznych zawiera opis wywołania MQCLOSE.

Opcje
Typ: MQLONG-input

Należy podać co najmniej jedną z następujących opcji:

- MQOO_BROWSE,
- MQOO_INPUT_ * (tylko jedna z tych)
- MQOO_INQUIRE
- MQOO_OUTPUT
- MQOO_SET
- MQOO_BIND_ * (tylko jedna z nich)

Szczegółowe informacje na temat tych opcji można znaleźć w poniższej tabeli. Inne opcje można określić zgodnie z wymaganiami. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe). Kombinacje, które nie są poprawne, są odnotowywane; wszystkie pozostałe kombinacje są poprawne. Dozwolone są tylko opcje mające zastosowanie do typu obiektu określonego przez parametr `ObjDesc` .

Tabela 553. Poprawne opcje MQOPEN dla kolejek i tematów

Opcja	Alias [1]	Lokalne i modelowe	Zdalna	Klaster nielokalny	Lista dystrybucyjna	Temat
MQOO_INPUT_AS_Q_DEF	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_INPUT_SHARED (mqoo_input_shared)	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_INPUT_EXCLUSIVE (mqoo_input_exclusive)	Tak	Tak	Nie	Nie	Nie	Nie
WYJŚCIA_Z_MQOT	Tak	Tak	Tak	Tak	Tak	Tak
MQOO_BROWSE (MQOO_PRZEGLĄDAJ)	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_CO_OP	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_INQUIRE (MQOO_INQUIRE)	Tak	Tak	[2]	Tak	Nie	Nie
MQOO_SET (zestaw MQOO_SET)	Tak	Tak	[2]	Nie	Nie	Nie
MQOO_BIND_ON_OPEN [3]	Tak	Tak	Tak	Tak	Tak	Nie
MQOO_BIND_NOT_FIXED [3]	Tak	Tak	Tak	Tak	Tak	Nie

Tabela 553. Poprawne opcje MQOPEN dla kolejek i tematów (kontynuacja)						
Opcja	**Alias** [1]	**Lokalne i modelowe**	**Zdalna**	**Klaster nielokalny**	**Lista dystrybucyjna**	**Temat**
MQOO_BIND_ON_GROUP [3]	Tak	Tak	Tak	Tak	Tak	Nie
MQOO_BIND_AS_Q_DEF [3]	Tak	Tak	Tak	Tak	Tak	Nie
MQOO_SAVE_ALL_CONTEXT	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_PASS_IDENTITY_CONTEXT (kontekst MQOO_PASS_IDENTITY_CONTEXT)	Tak	Tak	Tak	Tak	Tak	[4]
MQOO_PASS_ALL_CONTEXT (kontekst MQOO_PASS_ALL_CONTEXT)	Tak	Tak	Tak	Tak	Tak	Tak
MQOO_SET_IDENTITY_CONTEXT (MQOO_SET_IDENTITY_CONTEXT)	Tak	Tak	Tak	Tak	Tak	[4]
MQOO_SET_ALL_CONTEXT	Tak	Tak	Tak	Tak	Tak	Tak
MQOO_NO_READ_AHEAD (MQOO_NO_READ_AHEAD)	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_READ_AHEAD	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_READ_AHEAD_AS_Q_DEF	Tak	Tak	Nie	Nie	Nie	Nie
MQOO_ALTERNATE_USER_AUTHORITY	Tak	Tak	Tak	Tak	Tak	Tak
MQOO_FAIL_IF_QUIESCING (mqoo_fail_if_quiescing)	Tak	Tak	Tak	Tak	Tak	Tak
MQOO_RESOLVE_LOCAL_Q (MQOO_RESOLVE_LOCAL_Q)	Tak	Tak	Tak	Tak	Nie	Nie
MQOO_RESOLVE_LOCAL_TOPIC	Nie	Nie	Nie	Nie	Nie	Tak
MQOO_NO_MULTICAST (brak rozsyłania)	Nie	Nie	Nie	Nie	Nie	Tak

Uwagi:

1. Poprawność opcji dla aliasów zależy od poprawności opcji dla kolejki, na którą alias jest tłumaczony.
2. Ta opcja jest poprawna tylko dla lokalnej definicji kolejki zdalnej.
3. Tę opcję można określić dla dowolnego typu kolejki, ale jest ona ignorowana, jeśli kolejka nie jest kolejką klastra. Jednak atrybut kolejki **DefBind** nadpisuje kolejkę podstawową, nawet jeśli kolejka aliasowa nie znajduje się w klastrze.
4. Te atrybuty mogą być używane z tematem, ale mają wpływ tylko na kontekst ustawiony dla zachowanego komunikatu, a nie na pola kontekstu wysłane do dowolnego subskrybenta.

Opcje dostępu: Następujące opcje sterują typem operacji, które mogą być wykonywane na obiekcie:

MQOO_INPUT_AS_Q_DEF
Otwórz kolejkę, aby pobrać komunikaty przy użyciu wartości domyślnej zdefiniowanej przez kolejkę.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Typ dostępu jest współużytkowany lub wyłączny, w zależności od wartości atrybutu kolejki **DefInputOpenOption** . Szczegółowe informacje na ten temat zawiera sekcja “Atrybuty kolejek” na stronie 863 .

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_INPUT_SHARED
Otwórz kolejkę, aby pobrać komunikaty z dostępem współużytkowanym.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się powodzeniem, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację z opcją MQOO_INPUT_SHARED, ale kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta z opcją MQOO_INPUT_EXCLUSIVE.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_INPUT_EXCLUSIVE (wyłączna)
Otwórz kolejkę, aby pobrać komunikaty z dostępem na wyłączność.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację dla wejścia dowolnego typu (MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_OUTPUT

Otwórz kolejkę, aby umieścić komunikaty, lub temat lub łańcuch tematu, aby opublikować komunikaty.

Kolejka lub temat jest otwierany do użytku z kolejnymi wywołaniami MQPUT.

Wywołanie MQOPEN z tą opcją może zakończyć się pomyślnie, nawet jeśli atrybut kolejki **InhibitPut** jest ustawiony na wartość MQQA_PUT_INHIBITED (chociaż kolejne wywołania MQPUT kończą się niepowodzeniem, gdy atrybut jest ustawiony na tę wartość).

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych i tematów.

Do tych opcji mają zastosowanie następujące uwagi:

- Można podać tylko jedną z tych opcji.
- Wywołanie MQOPEN z jedną z tych opcji może zakończyć się pomyślnie, nawet jeśli atrybut kolejki **InhibitGet** jest ustawiony na wartość MQQA_GET_INHIBITED (chociaż kolejne wywołania MQGET kończą się niepowodzeniem, gdy atrybut jest ustawiony na tę wartość).
- Jeśli kolejka jest zdefiniowana jako niewspółużytkowalna (czyli atrybut kolejki **Shareability** ma wartość MQQA_NOT_SHAREABLE), próby otwarcia kolejki dla dostępu współużytkowanego są traktowane jako próby otwarcia kolejki z dostępem na wyłączność.
- Jeśli kolejka aliasowa jest otwarta z jedną z tych opcji, test do wyłącznego użycia (lub do tego, czy inna aplikacja ma wyłączne użycie) jest porównywany z kolejką podstawową, na którą alias jest tłumaczony.
- Te opcje nie są poprawne, jeśli **ObjectQMgrName** jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu **RemoteQMgrName** w lokalnej definicji kolejki zdalnej używanej na potrzeby określania aliasu menedżera kolejek jest nazwą lokalnego menedżera kolejek.

MQOO_BROWSE,
Otwórz kolejkę, aby przeglądać komunikaty.

Kolejka jest otwierana do użycia w kolejnych wywołaniach MQGET z jedną z następujących opcji:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR

Jest to dozwolone nawet wtedy, gdy kolejka jest obecnie otwarta dla opcji MQOO_INPUT_EXCLUSIVE. Wywołanie MQOPEN z opcją MQOO_BROWSE ustanawia kursor przeglądania i umieszcza go logicznie przed pierwszym komunikatem w kolejce. Więcej informacji na ten temat zawiera sekcja MQGMO-pole opcji .

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Nie jest ona również poprawna, jeśli `ObjectQMgrName` jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu **RemoteQMgrName** w definicji lokalnej kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

MQOO_CO_OP
Otwórz jako element współpracujący z zestawem uchwytów.

Ta opcja jest poprawna tylko z opcją MQOO_BROWSE. Jeśli ta opcja zostanie określona bez opcji MQOO_BROWSE, funkcja MQOPEN zwróci błąd MQRC_OPTIONS_ERROR.

Zwrócony uchwyt jest traktowany jako element współpracującego zestawu uchwytów dla kolejnych wywołań MQGET z jedną z następujących opcji:

- MQGMO_MARK_BROWSE_CO_OP
- MQGMO_UNMARKED_BROWSE_MSG
- MQGMO_UNMARK_BROWSE_CO_OP

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

MQOO_INQUIRE
Otwórz obiekt, aby uzyskać informacje o atrybutach.

Kolejka, lista nazw, definicja procesu lub menedżer kolejek są otwierane do użycia z kolejnymi wywołaniami MQINQ.

Ta opcja jest poprawna dla wszystkich typów obiektów innych niż listy dystrybucyjne. Nie jest on poprawny, jeśli parametr `ObjectQMgrName` jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu **RemoteQMgrName** w definicji lokalnej kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

MQOO_SET
Otwórz kolejkę, aby ustawić atrybuty.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQSET.

Ta opcja jest poprawna dla wszystkich typów kolejek innych niż listy dystrybucyjne. Nie jest ona poprawna, jeśli `ObjectQMgrName` jest nazwą lokalnej definicji kolejki zdalnej. Jest ona poprawna nawet wtedy, gdy wartość atrybutu **RemoteQMgrName** w lokalnej definicji kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

Opcje powiązania: Następujące opcje mają zastosowanie, gdy otwierany obiekt jest kolejką klastra. Te opcje sterują powiązaniem uchwytu kolejki z instancją kolejki klastra:

MQOO_BIND_ON_OPEN-OTWARTE
Lokalny menedżer kolejek wiąże uchwyt kolejki z instancją kolejki docelowej podczas otwierania kolejki. W rezultacie wszystkie komunikaty umieszczone za pomocą tego uchwytu są wysyłane do tej samej instancji kolejki docelowej i tej samej trasy.

Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja zostanie określona dla kolejki, która nie jest kolejką klastra, opcja zostanie zignorowana.

MQOO_BIND_NOT_FIXED (NIEUSTALONY)
Spowoduje to zatrzymanie lokalnego menedżera kolejek, który powiąże uchwyt kolejki z instancją kolejki docelowej. W wyniku tego kolejne wywołania MQPUT używające tego uchwytu wysyłają komunikaty do różnych instancji kolejki docelowej lub do tej samej instancji, ale przy użyciu różnych tras. Pozwala również na późniejszą zmianę wybranej instancji przez lokalny menedżer kolejek, przez zdalny menedżer kolejek lub przez agent kanału komunikatów (MCA) zgodnie z warunkami sieciowymi.

Uwaga: Aplikacje klienckie i serwerowe, które muszą wymieniać serię komunikatów w celu zakończenia transakcji, nie mogą używać opcji MQOO_BIND_NOT_FIXED (lub MQOO_BIND_AS_Q_DEF, gdy `DefBind` ma wartość MQBND_BIND_NOT_FIXED), ponieważ kolejne komunikaty w serii mogą być wysyłane do różnych instancji aplikacji serwera.

Jeśli dla kolejki klastra określono opcję MQOO_BROWSE lub jedną z opcji MQOO_INPUT_ *, menedżer kolejek musi wybrać lokalną instancję kolejki klastra. W rezultacie powiązanie uchwytu kolejki jest stałe, nawet jeśli podano parametr MQOO_BIND_NOT_FIXED.

Jeśli opcja MQOO_INQUIRE jest określona z opcją MQOO_BIND_NOT_FIXED, kolejne wywołania MQINQ używające tego uchwytu mogą wykonywać zapytania o różne instancje kolejki klastra, chociaż zwykle wszystkie instancje mają takie same wartości atrybutów.

Opcja MQOO_BIND_NOT_FIXED jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja zostanie określona dla kolejki, która nie jest kolejką klastra, opcja zostanie zignorowana.

MQOO_BIND_ON_GROUP
Umożliwia aplikacji żądanie przydzielenia grupy komunikatów do tej samej instancji docelowej.

Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja zostanie określona dla kolejki, która nie jest kolejką klastra, opcja zostanie zignorowana.

MQOO_BIND_AS_Q_DEF
Lokalny menedżer kolejek wiąże uchwyt kolejki w sposób zdefiniowany przez atrybut kolejki **DefBind** . Wartością tego atrybutu jest MQBND_BIND_ON_OPEN, MQBND_BIND_NOT_FIXED lub MQBND_BIND_ON_GROUP.

Opcja MQOO_BIND_AS_Q_DEF jest domyślna, jeśli nie określono opcji MQOO_BIND_ON_OPEN, MQOO_BIND_NOT_FIXED lub MQOO_BIND_ON_GROUP.

MQOO_BIND_AS_Q_DEF pomaga w dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną z pozostałych dwóch opcji wiązania, ale ponieważ jej wartość wynosi zero, nie można jej wykryć.

Opcje kontekstu: Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

MQOO_SAVE_ALL_CONTEXT
Informacje o kontekście są powiązane z tym uchwytem kolejki. Te informacje są ustawiane na podstawie kontekstu dowolnego komunikatu pobranego za pomocą tego uchwytu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Te informacje o kontekście można przekazać do komunikatu, który jest następnie umieszczany w kolejce przy użyciu wywołań MQPUT lub MQPUT1 . Patrz opcje MQPMO_PASS_IDENTITY_CONTEXT i MQPMO_PASS_ALL_CONTEXT opisane w sekcji “MQPMO-opcje umieszczania komunikatów” na stronie 514.

Dopóki komunikat nie zostanie pomyślnie pobrany, nie można przekazać kontekstu do komunikatu umieszczanego w kolejce.

Komunikat pobrany za pomocą jednej z opcji przeglądania MQGMO_BROWSE_ * nie ma zapisanych informacji o kontekście (chociaż pola kontekstu w parametrze **MsgDesc** są ustawiane po przeglądaniu).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Należy podać jedną z opcji MQOO_INPUT_ *.

MQOO_PASS_IDENTITY_CONTEXT,
Umożliwia to określenie opcji MQPMO_PASS_IDENTITY_CONTEXT w parametrze **PutMsgOpts** , gdy komunikat jest umieszczany w kolejce. W ten sposób komunikat otrzymuje informacje o kontekście tożsamości z kolejki wejściowej, która została otwarta za pomocą opcji MQOO_SAVE_ALL_CONTEXT. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

MQOO_PASS_ALL_CONTEXT
Umożliwia to określenie opcji MQPMO_PASS_ALL_CONTEXT w parametrze **PutMsgOpts** , gdy komunikat jest umieszczany w kolejce. W ten sposób komunikat otrzymuje informacje o tożsamości i kontekście źródłowym z kolejki wejściowej otwartej za pomocą opcji

MQOO_SAVE_ALL_CONTEXT. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Ta opcja implikuje kontekst MQOO_PASS_IDENTITY_CONTEXT, który nie musi być określony. Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

MQOO_SET_IDENTITY_CONTEXT,
Pozwala to na określenie opcji MQPMO_SET_IDENTITY_CONTEXT w parametrze **PutMsgOpts** , gdy komunikat jest umieszczany w kolejce. Daje to komunikatowi informacje o kontekście tożsamości zawarte w parametrze **MsgDesc** określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Ta opcja implikuje kontekst MQOO_PASS_IDENTITY_CONTEXT, który nie musi być określony. Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

MQOO_SET_ALL_CONTEXT
Umożliwia to określenie opcji MQPMO_SET_ALL_CONTEXT w parametrze **PutMsgOpts** , gdy komunikat jest umieszczany w kolejce. Daje to komunikatowi informacje o tożsamości i kontekście źródłowym zawarte w parametrze **MsgDesc** określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Ta opcja oznacza następujące opcje, które nie muszą być określone:

- MQOO_PASS_IDENTITY_CONTEXT,
- MQOO_PASS_ALL_CONTEXT
- MQOO_SET_IDENTITY_CONTEXT,

Należy określić opcję MQOO_OUTPUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

Opcje odczytu z wyprzedzeniem:

W przypadku wywołania MQOPEN z opcją MQOO_READ_AHEAD klient IBM MQ umożliwia odczyt z wyprzedzeniem, jeśli spełnione są pewne warunki. Są one następujące:

- Aplikacja kliencka musi zostać skompilowana i powiązana z wątkowymi bibliotekami klienta MQI IBM MQ.
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Następujące opcje określają, czy komunikaty nietrwałe są wysyłane do klienta przed żądaniem ich przez aplikację. Poniższe uwagi dotyczą opcji odczytu z wyprzedzeniem:

- Można podać tylko jedną z tych opcji.
- Te opcje są poprawne tylko dla kolejek lokalnych, aliasowych i modelowych. Nie są one poprawne dla kolejek zdalnych, list dystrybucyjnych, tematów ani menedżerów kolejek.
- Te opcje mają zastosowanie tylko wtedy, gdy określono jedną z następujących opcji: MQOO_BROWSE, MQOO_INPUT_SHARED i MQOO_INPUT_EXCLUSIVE, chociaż nie jest błędem określanie tych opcji za pomocą opcji MQOO_INQUIRE lub MQOO_SET.
- Jeśli aplikacja nie działa jako klient IBM MQ , te opcje są ignorowane.

MQOO_NO_READ_AHEAD
Komunikaty nietrwałe nie są wysyłane do klienta przed zażądaniem ich przez aplikację.

MQOO_READ_AHEAD
Komunikaty nietrwałe są wysyłane do klienta przed zażądaniem ich przez aplikację.

MQOO_READ_AHEAD_AS_Q_DEF
Zachowanie odczytu z wyprzedzeniem jest określane przez domyślny atrybut odczytu z wyprzedzeniem otwieranej kolejki. Jest to wartość domyślna.

Inne opcje: następujące opcje sterują sprawdzaniem autoryzacji, co się dzieje, gdy menedżer kolejek jest wyciszany, czy ma rozstrzygać nazwę kolejki lokalnej i rozsyłanie grupowe:

MQOO_ALTERNATE_USER_AUTHORITY (uprawnienie użytkownika na przemian)
Pole *AlternateUserId* w parametrze **ObjDesc** zawiera identyfikator użytkownika używany do sprawdzania poprawności tego wywołania MQOPEN. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy *AlternateUserId* ma uprawnienia do otwarcia obiektu z określonymi opcjami dostępu, bez względu na to, czy identyfikator użytkownika, pod którym działa aplikacja, jest do tego uprawniony. Nie ma to jednak zastosowania do określonych opcji kontekstu, które są zawsze sprawdzane pod kątem identyfikatora użytkownika, pod którym działa aplikacja.

Ta opcja jest poprawna dla wszystkich typów obiektów.

MQOO_FAIL_IF_QUIESCING,
Wywołanie MQOPEN nie powiedzie się, jeśli menedżer kolejek jest w stanie wyciszania.

z/OS W systemie z/OSw przypadku aplikacji w systemie CICS lub IMS ta opcja wymusza także niepowodzenie wywołania MQOPEN, jeśli połączenie jest w stanie wyciszania.

Ta opcja jest poprawna dla wszystkich typów obiektów.

Więcej informacji na temat kanałów klienta zawiera sekcja IBM MQ MQI clients.

MQOO_RESOLVE_LOCAL_Q
Wypełnij pole ResolvedQName w strukturze MQOD nazwą kolejki lokalnej, która została otwarta. Podobnie nazwa ResolvedQMgrjest wypełniana nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną. Jeśli struktura MQOD jest niższa niż wersja 3, opcja MQOO_RESOLVE_LOCAL_Q jest ignorowana i nie jest zwracany żaden błąd.

Kolejka lokalna jest zawsze zwracana, gdy otwarta jest kolejka lokalna, kolejka aliasowa lub kolejka modelowa, ale nie jest to możliwe, gdy na przykład kolejka zdalna lub kolejka nielokalnego klastra jest otwarta bez opcji MQOO_RESOLVE_LOCAL_Q; pola ResolvedQName i ResolvedQMgrNazwa są wypełnione nazwami RemoteQName i RemoteQMgr, które znajdują się w definicji kolejki zdalnej, lub w podobny sposób z wybraną zdalną kolejką klastra.

Jeśli podczas otwierania zostanie podana wartość MQOO_RESOLVE_LOCAL_Q, na przykład kolejka zdalna, ResolvedQName jest kolejką transmisji, do której są umieszczane komunikaty. Nazwa menedżera kolejek ResolvedQMgrjest wypełniana nazwą lokalnego menedżera kolejek udostępniającego kolejkę transmisji.

Jeśli użytkownik ma uprawnienia do przeglądania, wprowadzania lub wyprowadzania danych w kolejce, ma uprawnienia wymagane do określenia tej flagi w wywołaniu MQOPEN. Nie są wymagane żadne uprawnienia specjalne.

Ta opcja jest poprawna tylko dla kolejek i menedżerów kolejek.

MQOO_RESOLVE_LOCAL_TOPIC
Wypełnij pole ResolvedQName w strukturze MQOD, podając nazwę otwartego tematu administracyjnego.

MQOO_NO_MULTICAST (brak rozsyłania grupowego)
Komunikaty publikowania nie są wysyłane przy użyciu rozsyłania grupowego.

Ta opcja jest poprawna tylko z opcją MQOO_OUTPUT. Jeśli zostanie podany bez opcji MQOO_OUTPUT, funkcja MQOPEN zwróci wartość MQRC_OPTIONS_ERROR.

Ta opcja jest poprawna tylko dla tematu.

Hobj
Typ: MQHOBJ-wyjście

Ten uchwyt reprezentuje dostęp, który został ustanowiony dla obiektu. Musi być określona w kolejnych wywołaniach IBM MQ , które operują na obiekcie. Przestaje być ona poprawna po wywołaniu MQCLOSE lub po zakończeniu jednostki przetwarzania definiującej zasięg uchwytu.

Zasięg zwróconego uchwytu obiektu jest taki sam, jak zasięg uchwytu połączenia określonego w wywołaniu. Informacje na temat zasięgu uchwytu zawiera sekcja MQCONN-Hconn parameter (Parametr MQCONN-Hconn).

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_MULTIPLE_POWODY
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') Kolejka podstawowa aliasu nie jest poprawnym typem.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_NIEDOSTĘPNE
(2345, X' 929 ') Obiekt sprzęgania nie jest dostępny.

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') Sprawdzanie autoryzacji struktury narzędzia CF nie powiodło się.

MQRC_CF_STRUC_BŁĄD
(2349, X'92D') Niepoprawna struktura narzędzia CF.

MQRC_CF_STRUC_FAILED
(2373, X' 945 ') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Nagłówek listy struktury narzędzia CF w użyciu.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

BŁĄD MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

MQRC_CLUSTER_PUT_INHIBITED
(2268, X'8DC') Wywołania umieszczania zablokowane dla wszystkich kolejek w klastrze.

BŁĄD MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Rozstrzyganie nazwy klastra nie powiodło się.

BŁĄD ZASOBU MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Błąd zasobu klastra.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

BŁĄD MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') Domyślna kolejka transmisji nie jest lokalna.

MQRC_DEF_XMIT_Q_USAGE_ERROR,
(2199, X'897 ') Błąd użycia domyślnej kolejki transmisji.

BŁĄD MQRC_DYNAMIC_Q_NAME_ERROR
(2011, X'7DB') Niepoprawna nazwa kolejki dynamicznej.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') Brak dostępnych uchwytów.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

MQRC_MULTIPLE_POWODY
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

MQRC_NAME_IN_USE (nazwa użytkownika MQRC)
(2201, X'899 ') Nazwa w użyciu.

MQRC_NAME_NOT_VALID_FOR_TYPE (nazwa nie jest poprawna)
(2194, X'892 ') Nazwa obiektu jest niepoprawna dla typu obiektu.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

MQRC_OBJECT_ALREADY_EXISTS
(2100, X'834 ') Obiekt istnieje.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

MQRC_OBJECT_IN_UŻYTK
(2042, X'7FA') Obiekt jest już otwarty z opcjami powodującymi konflikt.

MQRC_OBJECT_LEVEL_NIEKOMPATYBILNY
(2360, X' 938 ') Poziom obiektu nie jest zgodny.

BŁĄD MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Niepoprawna nazwa obiektu.

MQRC_OBJECT_NOT_UNIQUE (obiekt MQRC_NOT_UNIQUE)
(2343, X' 927 ') Obiekt nie jest unikalny.

Błąd MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Niepoprawna nazwa menedżera kolejek obiektu.

BŁĄD REKORDÓW MQRC_OBJECT_RECORDS
(2155, X'86B') Rekordy obiektów są niepoprawne.

BŁĄD WYWOŁANIA MQRC_OBJECT_STRING_ERROR
(2441, X'0989 ') Niepoprawne pole Objectstring

BŁĄD MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Niepoprawny typ obiektu.

BŁĄD OD_MQRC
(2044, X'7FC') Niepoprawna struktura deskryptora obiektu.

MQRC_OPTION_NOT_VALID_FOR_TYPE
(2045, X'7FD') Opcja niepoprawna dla typu obiektu.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_PAGESET_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

BŁĄD MQRC_Q_TYPE_ERROR
(2057, X'809 ') Niepoprawny typ kolejki.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Niepoprawna liczba rekordów.

BŁĄD NAZWY MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') Niepoprawna nazwa kolejki zdalnej.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Rekordy odpowiedzi są niepoprawne.

BŁĄD MQRC_SECURITY_ERROR
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

BŁĄD SYNTAX_MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Wywołanie MQOPEN, MQPUT1 lub MQSUB zostało wykonane, ale określono instrukcję wyboru, która zawierała błąd składniowy.

MQRC_STOPPED_BY_CLUSTER_EXIT,
(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Nieznana kolejka podstawowa aliasów.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Nieznana domyślna kolejka transmisji.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nieznana nazwa obiektu.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Nieznany menedżer kolejek obiektów.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Nieznany zdalny menedżer kolejek.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Nieznana kolejka transmisji.

MQRC_WRONG_CF_LEVEL
(2366, X'93E') Struktura narzędzia CF jest na niewłaściwym poziomie.

BŁĄD MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Kolejka transmisji nie jest lokalna.

MQRC_XMIT_Q_USAGE_ERROR,
(2092, X'82C') Kolejka transmisji z niewłaściwym użyciem.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Ogólne uwagi dotyczące użycia

1. Otwarty obiekt jest jednym z następujących obiektów:
   - Kolejka do:
     - Pobieranie lub przeglądanie komunikatów (za pomocą wywołania MQGET)
     - Komunikaty typu put (przy użyciu wywołania MQPUT)
     - Uzyskiwanie informacji o atrybutach kolejki (za pomocą wywołania MQINQ)
     - Ustawianie atrybutów kolejki (za pomocą wywołania MQSET)

     Jeśli kolejka o nazwie jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Patrz opis parametru **ObjDesc** w sekcji “MQOPEN-otwarcie obiektu” na stronie 756.

     Lista dystrybucyjna jest specjalnym typem obiektu kolejki, który zawiera listę kolejek. Można go otwierać w celu umieszczania komunikatów, ale nie w celu pobierania lub przeglądania komunikatów, a także w celu uzyskiwania informacji lub ustawiania atrybutów. Więcej informacji na ten temat zawiera uwaga 8.

     Kolejka o typie `QSGDISP(GROUP)` jest specjalnym typem definicji kolejki, który nie może być używany z wywołaniami MQOPEN lub MQPUT1 .
   - Lista nazw do odpytywania o nazwy kolejek na liście (przy użyciu wywołania MQINQ).
   - Definicja procesu do uzyskiwania informacji o atrybutach procesu (za pomocą wywołania MQINQ).

- Menedżer kolejek, który ma uzyskać informacje o atrybutach lokalnego menedżera kolejek (przy użyciu wywołania MQINQ).
- Temat do publikowania komunikatu (przy użyciu wywołania MQPUT)

2. Aplikacja może otworzyć ten sam obiekt więcej niż jeden raz. Dla każdego otwarcia zwracany jest inny uchwyt obiektu. Każdy zwrócony uchwyt może być użyty dla funkcji, dla których wykonano odpowiednie otwarcie.
3. Jeśli otwierany obiekt jest kolejką inną niż kolejka klastra, wszystkie tłumaczenia nazw w menedżerze kolejek lokalnych są wykonywane w czasie wywołania MQOPEN. Może to obejmować:
   - Tłumaczenie nazwy lokalnej definicji kolejki zdalnej na nazwę zdalnego menedżera kolejek oraz nazwy, pod jaką kolejka jest znana w zdalnym menedżerze kolejek
   - Tłumaczenie nazwy menedżera kolejek zdalnych na nazwę lokalnej kolejki transmisji
   - z/OS Tylko w systemie z/OS : tłumaczenie nazwy zdalnego menedżera kolejek na nazwę współużytkowanej kolejki transmisji używanej przez agenta IGQ (ma zastosowanie tylko wtedy, gdy lokalne i zdalne menedżery kolejek należą do tej samej grupy współużytkowania kolejek)
   - Tłumaczenie aliasu na nazwę kolejki podstawowej lub obiektu tematu.

   Należy jednak pamiętać, że kolejne wywołania MQINQ lub MQSET dla uchwytu odnoszą się wyłącznie do nazwy, która została otwarta, a nie do obiektu będącego wynikiem translacji nazw. Na przykład, jeśli otwarty obiekt jest aliasem, atrybuty zwracane przez wywołanie MQINQ są atrybutami aliasu, a nie atrybutami kolejki podstawowej lub obiektu tematu, na który alias jest tłumaczony.

   Jeśli otwierany obiekt jest kolejką klastra, tłumaczenie nazw może wystąpić w czasie wywołania MQOPEN lub może zostać odroczone do czasu późniejszego. Punkt, w którym występuje rozstrzyganie, jest sterowany przez opcje MQOO_BIND_ * określone w wywołaniu MQOPEN:
   - MQOO_BIND_ON_OPEN-OTWARTE
   - MQOO_BIND_NOT_FIXED (NIEUSTALONY)
   - MQOO_BIND_AS_Q_DEF
   - MQOO_BIND_ON_GROUP

   Więcej informacji na temat tłumaczenia nazw dla kolejek klastra zawiera sekcja Rozstrzyganie nazw .
4. Wywołanie MQOPEN z opcją MQOO_BROWSE ustanawia kursor przeglądania do użycia z wywołaniami MQGET, które określają uchwyt obiektu i jedną z opcji przeglądania. Umożliwia to skanowanie kolejki bez zmiany jej zawartości. Komunikat znaleziony podczas przeglądania można usunąć z kolejki za pomocą opcji MQGMO_MSG_UNDER_CURSOR.

   Wiele kursorów przeglądania może być aktywnych dla jednej aplikacji, wysyłając kilka żądań MQOPEN dla tej samej kolejki.
5. Do aplikacji uruchamianych przez monitor wyzwalacza jest przekazywana nazwa kolejki, która jest powiązana z aplikacją podczas jej uruchamiania. Tę nazwę kolejki można podać w parametrze **ObjDesc** , aby ją otworzyć. Więcej informacji na ten temat zawiera sekcja “MQTMC2 -komunikat wyzwalacza 2 (format znakowy)” na stronie 625.

Opcje odczytu z wyprzedzeniem

W przypadku wywołania MQOPEN z opcją MQOO_READ_AHEAD klient IBM MQ umożliwia odczyt z wyprzedzeniem, jeśli spełnione są pewne warunki. Są one następujące:

- Aplikacja kliencka musi zostać skompilowana i powiązana z wątkowymi bibliotekami klienta MQI IBM MQ.
- Kanał klienta musi używać protokołu TCP/IP.
- Ustawienie SharingConversations (SHARECNV) kanału musi mieć wartość niezerową w definicji kanału zarówno klienta, jak i serwera.

Poniższe uwagi dotyczą korzystania z opcji odczytu z wyprzedzeniem.

1. Opcje odczytu z wyprzedzeniem mają zastosowanie tylko wtedy, gdy określono jedną i tylko jedną z opcji MQOO_BROWSE, MQOO_INPUT_SHARED i MQOO_INPUT_EXCLUSIVE. Błąd nie jest zgłaszany, jeśli opcje odczytu z wyprzedzeniem zostały określone z opcjami MQOO_ INQUIRE lub MQOO_SET.
2. Odczyt z wyprzedzeniem nie jest włączany, jeśli opcje użyte w pierwszym wywołaniu MQGET nie są obsługiwane w przypadku odczytu z wyprzedzeniem. Odczyt z wyprzedzeniem jest również wyłączony, gdy klient łączy się z menedżerem kolejek, który nie obsługuje odczytu z wyprzedzeniem.
3. Jeśli aplikacja nie działa jako klient IBM MQ , opcje odczytu z wyprzedzeniem są ignorowane.

Kolejki klastra

Poniższe uwagi dotyczą użycia kolejek klastra.

1. Jeśli kolejka klastra jest otwierana po raz pierwszy, a menedżer kolejek lokalnych nie jest menedżerem kolejek repozytorium pełnego, menedżer kolejek lokalnych uzyskuje informacje o kolejce klastra z menedżera kolejek repozytorium pełnego. Jeśli sieć jest zajęta, może upłynąć kilka sekund, nim lokalny menedżer kolejek odbierze potrzebne informacje z menedżera kolejek repozytorium. W wyniku tego aplikacja wydająca wywołanie MQOPEN może oczekiwać do 10 sekund, zanim sterowanie wróci z wywołania MQOPEN. Jeśli w tym czasie lokalny menedżer kolejek nie otrzyma potrzebnych informacji o kolejce klastra, wywołanie zakończy się niepowodzeniem z kodem przyczyny MQRC_CLUSTER_RESOLUTION_ERROR.
2. Jeśli kolejka klastra jest otwarta i istnieje wiele instancji kolejki w klastrze, otwarta instancja zależy od opcji określonych w wywołaniu MQOPEN:
   - Jeśli podane opcje obejmują jedną z następujących opcji:
     - MQOO_BROWSE,
     - MQOO_INPUT_AS_Q_DEF
     - MQOO_INPUT_EXCLUSIVE (wyłączna)
     - MQOO_INPUT_SHARED
     - MQOO_SET

     Instancja otwartej kolejki klastra musi być instancją lokalną. Jeśli nie ma lokalnej instancji kolejki, wywołanie MQOPEN nie powiedzie się.
   - Jeśli podane opcje nie zawierają żadnej z opcji opisanych wcześniej, ale zawierają jedną lub obie z następujących opcji:
     - MQOO_INQUIRE
     - MQOO_OUTPUT

     instancja otwarta jest instancją lokalną (jeśli istnieje), a instancja zdalna-w przeciwnym razie (jeśli używane są wartości domyślne CLWLUSEQ). Instancja wybrana przez menedżera kolejek może jednak zostać zmieniona przez wyjście obciążenia klastra (jeśli istnieje).
3. Jeśli istnieje subskrypcja dla kolejki, ale nie jest ona potwierdzona przez pełne repozytorium, obiekt nie jest obecny w klastrze i wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_NAME.

Więcej informacji na temat kolejek klastrów zawiera sekcja Kolejki klastrów.

Lista dystrybucyjna

Poniższe uwagi dotyczą użycia list dystrybucyjnych.

Listy dystrybucyjne są obsługiwane w następujących środowiskach:

- AIX
- IBM i
- Linux

- Windows Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

1. Podczas otwierania listy dystrybucyjnej pola w strukturze MQOD muszą być ustawione w następujący sposób:
   - `Version` musi mieć wartość MQOD_VERSION_2 lub większą.
   - Parametr `ObjectType` musi mieć wartość MQOT_Q.
   - Parametr `ObjectName` musi być pusty lub musi być łańcuchem o wartości NULL.
   - Parametr `ObjectQMgrName` musi być pusty lub musi być łańcuchem o wartości NULL.
   - Wartość `RecsPresent` musi być większa od zera.
   - Jeden z `ObjectRecOffset` i `ObjectRecPtr` musi być zerem, a drugi niezerem.
   - Nie więcej niż jedna z wartości `ResponseRecOffset` i `ResponseRecPtr` może być niezerowa.
   - Muszą istnieć rekordy obiektów `RecsPresent` , adresowane przez `ObjectRecOffset` lub `ObjectRecPtr`. Rekordy obiektów muszą być ustawione na nazwy kolejek docelowych, które mają zostać otwarte.
   - Jeśli jeden z `ResponseRecOffset` i `ResponseRecPtr` jest niezerowy, muszą istnieć rekordy odpowiedzi `RecsPresent` . Są one ustawiane przez menedżer kolejek, jeśli wywołanie zostanie zakończone z kodem przyczyny MQRC_MULTIPLE_REASON.

   Za pomocą programu MQOD w wersji version-2 można również otworzyć pojedynczą kolejkę, która nie znajduje się na liście dystrybucyjnej, upewniając się, że parametr `RecsPresent` ma wartość zero.
2. W parametrze **`Options`** poprawne są tylko następujące opcje otwierania:
   - MQOO_OUTPUT
   - MQOO_PASS_ * _KONTEKST
   - MQOO_SET_ * _XX_ENCODE_CASE_ONE kontekst
   - MQOO_ALTERNATE_USER_AUTHORITY (uprawnienie użytkownika na przemian)
   - MQOO_FAIL_IF_QUIESCING,
3. Kolejki docelowe na liście dystrybucyjnej mogą być kolejkami lokalnymi, aliasowymi lub zdalnymi, ale nie mogą być kolejkami modelowymi. Jeśli określono kolejkę modelową, otwarcie tej kolejki nie powiedzie się z kodem przyczyny MQRC_Q_TYPE_ERROR. Nie uniemożliwia to jednak pomyślnego otwarcia innych kolejek na liście.
4. Parametry kodu zakończenia i kodu przyczyny są ustawione w następujący sposób:
   - Jeśli wszystkie operacje otwarcia kolejek na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, parametry kodu zakończenia i kodu przyczyny są ustawione w taki sposób, aby opisywać wspólny wynik. W tym przypadku rekordy odpowiedzi MQRR (jeśli są udostępniane przez aplikację) nie są ustawione.

     Na przykład, jeśli każde otwarcie powiedzie się, kod zakończenia zostanie ustawiony na MQCC_OK, a kod przyczyny na MQRC_NONE; jeśli każde otwarcie nie powiedzie się, ponieważ żadna z kolejek nie istnieje, parametry zostaną ustawione na MQCC_FAILED i MQRC_UNKNOWN_OBJECT_NAME.
   - Jeśli operacje otwarcia dla kolejek na liście dystrybucyjnej nie powiodą się lub zakończą się niepowodzeniem w ten sam sposób:
     - Parametr kodu zakończenia jest ustawiany na wartość MQCC_WARNING, jeśli co najmniej jedno otwarcie zakończyło się pomyślnie, lub na wartość MQCC_FAILED, jeśli wszystkie nie powiodły się.
     - Parametr kodu przyczyny jest ustawiony na wartość MQRC_MULTIPLE_REASON.
     - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.
5. Jeśli lista dystrybucyjna została pomyślnie otwarta, uchwyt `Hobj` zwrócony przez wywołanie może być używany w kolejnych wywołaniach MQPUT do umieszczania komunikatów w kolejkach na liście

dystrybucyjnej, a w wywołaniu MQCLOSE do zrzeczenia się dostępu do listy dystrybucyjnej. Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest MQCO_NONE.

Wywołania MQPUT1 można również użyć do umieszczenia komunikatu na liście dystrybucyjnej. Struktura MQOD definiująca kolejki na liście jest określona jako parametr w tym wywołaniu.

6. Każde pomyślnie otwarte miejsce docelowe na liście dystrybucyjnej jest traktowane jako oddzielny uchwyt podczas sprawdzania, czy aplikacja przekroczyła maksymalną dozwoloną liczbę uchwytów (patrz atrybut menedżera kolejek systemu **MaxHandles**). Ma to miejsce nawet wtedy, gdy co najmniej dwa miejsca docelowe na liście dystrybucyjnej są tłumaczone na tę samą kolejkę fizyczną. Jeśli wywołanie MQOPEN lub MQPUT1 dla listy dystrybucyjnej spowodowałoby przekroczenie liczby uchwytów używanych przez aplikację `MaxHandles`, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_HANDLE_NOT_AVAILABLE.
7. Każde pomyślnie otwarte miejsce docelowe ma wartość atrybutu **OpenOutputCount** zwiększoną o jeden. Jeśli dwa lub więcej miejsc docelowych na liście dystrybucyjnej jest rozstrzyganych do tej samej kolejki fizycznej, to atrybut **OpenOutputCount** tej kolejki jest zwiększany o liczbę miejsc docelowych na liście dystrybucyjnej, które są rozstrzygane do tej kolejki.
8. Każda zmiana definicji kolejek, która spowodowałaby, że uchwyt stałby się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana w ścieżce rozstrzygania), nie powoduje, że uchwyt listy dystrybucyjnej stałby się niepoprawny. Jednak powoduje to niepowodzenie dla tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnym wywołaniu MQPUT.
9. Lista dystrybucyjna może zawierać tylko jedno miejsce docelowe.

Kolejki zdalne

Poniższe uwagi dotyczą użycia kolejek zdalnych.

Kolejkę zdalną można określić na jeden z dwóch sposobów w parametrze **ObjDesc** tego wywołania.

- Przez określenie dla `ObjectName` nazwy lokalnej definicji kolejki zdalnej. W tym przypadku parametr `ObjectQMgrName` odwołuje się do lokalnego menedżera kolejek i może zostać określony jako łańcuch pusty lub (w języku programowania C) jako łańcuch pusty.

  Sprawdzenie poprawności zabezpieczeń wykonywane przez menedżer kolejek lokalnych sprawdza, czy użytkownik ma uprawnienia do otwarcia lokalnej definicji kolejki zdalnej.
- Przez określenie dla parametru `ObjectName` nazwy zdalnej kolejki znanej zdalnemu menedżerowi kolejek. W tym przypadku `ObjectQMgrName` jest nazwą zdalnego menedżera kolejek.

  Sprawdzanie poprawności zabezpieczeń wykonywane przez menedżer kolejek lokalnych sprawdza, czy użytkownik jest autoryzowany do wysyłania komunikatów do kolejki transmisji w wyniku procesu tłumaczenia nazw.

W obu przypadkach:

- Lokalny menedżer kolejek nie wysyła żadnych komunikatów do zdalnego menedżera kolejek, aby sprawdzić, czy użytkownik ma uprawnienia do umieszczania komunikatów w kolejce.
- Po nadejściu komunikatu do zdalnego menedżera kolejek zdalny menedżer kolejek może go odrzucić, ponieważ użytkownik, który zainicjuje komunikat, nie jest autoryzowany.

Więcej informacji na ten temat zawierają pola `ObjectName` i `ObjectQMgrName` opisane w sekcji “MQOD-deskryptor obiektu” na stronie 493 .

Obiekty

Zabezpieczenia

Poniższe uwagi dotyczą aspektów bezpieczeństwa związanych z używaniem komendy MQOPEN.

Menedżer kolejek przeprowadza sprawdzanie zabezpieczeń podczas wykonywania wywołania MQOPEN w celu sprawdzenia, czy identyfikator użytkownika, pod którym działa aplikacja, ma odpowiedni poziom

uprawnień, zanim dostęp będzie dozwolony. Sprawdzanie uprawnień jest wykonywane dla nazwy otwieranego obiektu, a nie dla nazwy lub nazw, które są wynikiem translacji nazwy.

Jeśli otwierany obiekt jest kolejką aliasową, która wskazuje obiekt tematu, menedżer kolejek przeprowadza sprawdzenie zabezpieczeń nazwy kolejki aliasowej przed wykonaniem sprawdzenia zabezpieczeń tematu, tak jakby obiekt tematu był używany bezpośrednio.

Jeśli otwierany obiekt jest obiektem tematu, niezależnie od tego, czy jest to obiekt `ObjectName` , czy obiekt `ObjectString` (z bazowym obiektem `ObjectName`lub bez niego), menedżer kolejek wykonuje sprawdzanie zabezpieczeń przy użyciu wynikowego łańcucha tematu pobranego z obiektu tematu określonego w temacie `ObjectName`, a jeśli jest to wymagane, konkatenując go z obiektem udostępnionym w produkcie `ObjectString`, a następnie znajdując najbliższy obiekt tematu w danym punkcie drzewa tematów lub powyżej, aby wykonać sprawdzenie zabezpieczeń. Może to nie być ten sam obiekt tematu, który został określony w pliku `ObjectName`.

Jeśli otwierany obiekt jest kolejką modelową, menedżer kolejek wykonuje pełne sprawdzenie zabezpieczeń zarówno w odniesieniu do nazwy kolejki modelowej, jak i nazwy kolejki dynamicznej, która została utworzona. Jeśli wynikowa kolejka dynamiczna zostanie otwarta jawnie, zostanie wykonane dodatkowe sprawdzenie bezpieczeństwa zasobu względem nazwy kolejki dynamicznej.

z/OS W systemie z/OSmenedżer kolejek wykonuje sprawdzenia zabezpieczeń tylko wtedy, gdy zabezpieczenia są włączone. Więcej informacji na temat sprawdzania zabezpieczeń zawiera sekcja Konfigurowanie zabezpieczeń w systemie z/OS .

Atrybuty

Poniższe uwagi odnoszą się do atrybutów.

Atrybuty obiektu mogą się zmieniać, gdy aplikacja ma otwarty obiekt. W wielu przypadkach aplikacja nie zauważa tego, ale w przypadku niektórych atrybutów menedżer kolejek oznacza uchwyt jako niepoprawny. Są to następujące atrybuty:

- Dowolny atrybut, który ma wpływ na tłumaczenie nazw obiektu. Ma to zastosowanie niezależnie od użytych opcji otwierania i obejmuje następujące elementy:
  - Zmiana atrybutu **BaseQName** kolejki aliasowej, która jest otwarta.
  - Zmiana atrybutu **TargetType** kolejki aliasowej, która jest otwarta.
  - Zmiana atrybutów kolejki **RemoteQName** lub **RemoteQMgrName** dla dowolnego uchwytu otwartego dla tej kolejki lub dla kolejki rozstrzyganej za pomocą tej definicji jako alias menedżera kolejek.
  - Każda zmiana, która powoduje, że aktualnie otwarty uchwyt kolejki zdalnej jest tłumaczony na inną kolejkę transmisji lub nie udaje się w ogóle przetłumaczyć na jedną. Na przykład może to być:
    - Zmiana atrybutu **XmitQName** lokalnej definicji kolejki zdalnej, bez względu na to, czy definicja jest używana dla kolejki, czy dla aliasu menedżera kolejek.
    - z/OS Tylko w systemie z/OS : zmiana wartości atrybutu menedżera kolejek systemu **IntraGroupqueuing** lub zmiana definicji współużytkowanej kolejki transmisji (SYSTEM.QSG.TRANSMIT.QUEUE) używany przez agenta IGQ.

    Istnieje jeden wyjątek: utworzenie nowej kolejki transmisji. Uchwyt, który zostałby przetłumaczony na tę kolejkę, gdyby był obecny podczas otwierania uchwytu, ale zamiast tego został przetłumaczony na domyślną kolejkę transmisji, nie jest niepoprawny.
  - Zmiana atrybutu **DefXmitQName** menedżera kolejek. W tym przypadku wszystkie otwarte uchwyty, które zostały przetłumaczone na poprzednio nazwaną kolejkę (które zostały przetłumaczone tylko dlatego, że była to domyślna kolejka transmisji), są oznaczone jako niepoprawne. Nie ma to wpływu na uchwyty, które zostały przetłumaczone na tę kolejkę z innych przyczyn.
- Atrybut kolejki **Shareability** , jeśli istnieją co najmniej dwa uchwyty, które obecnie zapewniają dostęp MQOO_INPUT_SHARED dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę. Jeśli tak, wszystkie uchwyty otwarte dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę, są oznaczane jako niepoprawne, niezależnie od opcji otwarcia.

z/OS W systemie z/OSopisane wcześniej uchwyty są oznaczone jako niepoprawne, jeśli co najmniej jeden uchwyt zapewnia obecnie dostęp do kolejki na poziomie MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE.

- Atrybut kolejki **Usage** dla wszystkich uchwytów otwartych dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę, niezależnie od opcji otwarcia.

Jeśli uchwyt jest oznaczony jako niepoprawny, wszystkie kolejne wywołania (inne niż MQCLOSE) używające tego uchwytu kończą się niepowodzeniem z kodem przyczyny MQRC_OBJECT_CHANGED. Aplikacja musi wydać wywołanie MQCLOSE (przy użyciu oryginalnego uchwytu), a następnie ponownie otworzyć kolejkę. Wszystkie niezatwierdzone aktualizacje dotyczące starego uchwytu z poprzednich pomyślnych wywołań mogą być nadal zatwierdzone lub wycofane, zgodnie z wymaganiami logiki aplikacji.

Jeśli taka sytuacja ma miejsce w przypadku zmiany atrybutu, należy użyć specjalnej wersji wymuszenia wywołania.

Wywołanie C

```
MQOPEN (Hconn, &ObjDesc, Options, &Hobj, &CompCode,
       &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQOPEN' USING HCONN, OBJDESC, OPTIONS, HOBJ, COMPCODE, REASON
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQOPEN (Hconn, ObjDesc, Options, Hobj, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQOPEN,(HCONN,OBJDESC,OPTIONS,HOBJ,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

Windows

```
MQOPEN Hconn, ObjDesc, Options, Hobj, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-komunikat umieszczania

Wywołanie MQPUT umieszcza komunikat w kolejce, na liście dystrybucyjnej lub w temacie. Kolejka, lista dystrybucyjna lub temat muszą być już otwarte.

Składnia

MQPUT (*Hconn*, *Hobj*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Reason*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość Hconn została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

z/OS W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Hobj

Typ: MQHOBJ-input

Ten uchwyt reprezentuje kolejkę, do której komunikat jest dodawany, lub temat, do którego komunikat jest publikowany. Wartość Hobj została zwrócona przez poprzednie wywołanie MQOPEN, które określało opcję MQOO_OUTPUT.

MsgDesc
Typ: MQMD-wejście/wyjście

Ta struktura opisuje atrybuty wysyłanego komunikatu i odbiera informacje o komunikacie po zakończeniu żądania umieszczenia. Szczegółowe informacje można znaleźć w sekcji "MQMD-deskryptor komunikatu" na stronie 432.

Jeśli aplikacja udostępnia strukturę MQMD version-1 , dane komunikatu można poprzedzić strukturą MQMDE, aby określić wartości pól, które istnieją w strukturze MQMD version-2 , ale nie w strukturze version-1. Pole *Format* w deskryptorze MQMD musi być ustawione na wartość MQFMT_MD_EXTENSION, aby wskazać, że istnieje MQMDE. Więcej informacji na temat zawiera sekcja "MQMDE-rozszerzenie deskryptora komunikatu" na stronie 485.

Aplikacja nie musi udostępniać struktury MQMD, jeśli w polach `OriginalMsgHandle` lub `NewMsgHandle` struktury MQPMO podano poprawny uchwyt komunikatu. Jeśli w jednym z tych pól nie zostanie podana żadna wartość, deskryptor komunikatu jest pobierany z deskryptora powiązanego z uchwytami komunikatów.

Jeśli używane lub planowane jest użycie wyjść funkcji API, zaleca się jawne podanie struktury MQMD i nieużywanie deskryptorów komunikatów powiązanych z uchwytami komunikatów. Jest to spowodowane tym, że wyjście funkcji API powiązane z wywołaniem MQPUT lub MQPUT1 nie może określić, które wartości MQMD są używane przez menedżer kolejek do wykonania żądania MQPUT lub MQPUT1 .

PutMsg-opcje
Typ: MQPMO-wejście/wyjście

Szczegółowe informacje można znaleźć w sekcji "MQPMO-opcje umieszczania komunikatów" na stronie 514.

BufferLength
Typ: MQLONG-input

Długość komunikatu w pliku `Buffer`. Zero jest poprawne i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit dla `BufferLength` zależy od różnych czynników:

- Jeśli miejsce docelowe jest kolejką lokalną lub jest tłumaczone na kolejkę lokalną, górny limit zależy od tego, czy:
  - Lokalny menedżer kolejek obsługuje segmentację.
  - Aplikacja wysyłająca określa flagę, która umożliwia menedżerowi kolejek segmentowanie komunikatu. Ta opcja ma wartość MQMF_SEGMENTATION_ALLOWED i można ją określić w MQMD version-2 lub w MQMDE używanym z MQMD version-1 .

  Jeśli oba te warunki są spełnione, wartość `BufferLength` nie może przekroczyć 999 999 999 minus wartość pola `Offset` w strukturze MQMD. Najdłuższy komunikat logiczny, który można umieścić, to 999 999 999 bajtów (jeśli parametr `Offset` ma wartość zero). Jednak ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym działa aplikacja, mogą spowodować obniżenie limitu.

  Jeśli jeden lub oba powyższe warunki nie są spełnione, wartość `BufferLength` nie może przekroczyć mniejszej wartości atrybutu **MaxMsgLength** kolejki i atrybutu **MaxMsgLength** menedżera kolejek.
- Jeśli miejsce docelowe jest kolejką zdalną lub jest tłumaczone na kolejkę zdalną, mają zastosowanie warunki dla kolejek lokalnych, ale dla każdego menedżera kolejek, przez który komunikat musi przejść, aby osiągnąć kolejkę docelową. w szczególności:
  1. Lokalna kolejka transmisji używana do tymczasowego przechowywania komunikatu w menedżerze kolejek lokalnych
  2. Pośrednie kolejki transmisji (jeśli istnieją) używane do przechowywania komunikatu w menedżerach kolejek na trasie między lokalnymi i docelowymi menedżerami kolejek
  3. Kolejka docelowa w menedżerze kolejek docelowych

Najdłuższy komunikat, który można umieścić, jest więc zarządzany przez najbardziej restrykcyjne z tych kolejek i menedżerów kolejek.

Gdy komunikat znajduje się w kolejce transmisji, dodatkowe informacje rezydują z danymi komunikatu, co zmniejsza ilość danych aplikacji, które mogą być przenoszone. W takiej sytuacji podczas określania limitu dla `BufferLength`należy odjąć bajty MQ_MSG_HEADER_LENGTH od wartości `MaxMsgLength` kolejek transmisji.

Uwaga: Tylko niespełnienie warunku 1 może być diagnozowane synchronicznie (z kodem przyczyny MQRC_MSG_TOO_BIG_FOR_Q lub MQRC_MSG_TOO_BIG_FOR_Q_MGR) podczas umieszczania komunikatu. Jeśli warunki 2 lub 3 nie są spełnione, komunikat jest przekierowywany do kolejki niedostarczonych komunikatów (niedostarczonych komunikatów) w pośrednim menedżerze kolejek lub w docelowym menedżerze kolejek. W takim przypadku generowany jest komunikat raportu, jeśli został on zażądany przez nadawcę.

Buforuj
Typ: MQBYTExBuffer-długość-wejście

Jest to bufor zawierający dane aplikacji do wysłania. Bufor musi być wyrównany do granicy odpowiedniej dla rodzaju danych w komunikacie. Wyrównanie 4-bajtowe jest odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówków IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli plik `Buffer` zawiera dane znakowe lub liczbowe, należy ustawić pola `CodedCharSetId` i `Encoding` w parametrze **`MsgDesc`** na wartości odpowiednie dla danych. Umożliwia to odbiorcy komunikatu przekształcanie danych (w razie potrzeby) na zestaw znaków i kodowanie używane przez odbiorcę.

Uwaga: Wszystkie inne parametry w wywołaniu MQPUT muszą mieć zestaw znaków i kodowanie lokalnego menedżera kolejek (określone przez atrybut menedżera kolejek **`CodedCharSetId`** i parametr MQENC_NATIVE).

W języku programowania C parametr jest zadeklarowany jako wskaźnik do unieważnienia; jako parametr można podać adres dowolnego typu danych.

Jeśli parametr **`BufferLength`** ma wartość zero, nie jest przywoływany parametr `Buffer` ; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość null.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako `CompCode`.

Jeśli `CompCode` ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli `CompCode` ma wartość MQCC_WARNING:

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889 ') Niespójna specyfikacja trwałości.

MQRC_INCONSISTENT_UOW,
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_MULTIPLE_POWODY
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

MQRC_PRIORITY_PRZEKROCZONE_MAKSIMUM
(2049, X'801 ') Priorytet komunikatu przekracza maksymalną obsługiwaną wartość.

MQRC_UNKNOWN_REPORT_OPCJA
(2104, X'838 ') Nierozpoznane opcje raportu w deskryptorze komunikatu.

Jeśli CompCode ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ALIAS_TARGTYPE_CHANGED
(2480, X'09B0') Typ celu subskrypcji został zmieniony z kolejki na obiekt tematu.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BACKED_OUT
(2003, X'7D3') Jednostka pracy wycofała się.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'7D4') Niepoprawny parametr buforu.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CALL_PRZERWANE
(2549, X'9F5') MQPUT lub MQCMIT zostało przerwane i przetwarzanie ponownego połączenia nie może ponownie ustanowić określonego wyniku.

MQRC_CF_NIEDOSTĘPNE
(2345, X' 929 ') Obiekt sprzęgania nie jest dostępny.

MQRC_CF_STRUC_FAILED
(2373, X' 945 ') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

BŁĄD MQRC_CFGR_ERROR
(2416, X' 970 ') Struktura parametru grupy PCF MQCFGR w danych komunikatu jest niepoprawna.

BŁĄD MQRC_CFH_ERROR
(2235, X'8BB') Niepoprawna struktura nagłówka PCF.

MQRC_CFIF_BŁĄD
(2414, X'96E') Struktura parametru filtru liczby całkowitej PCF w danych komunikatu jest niepoprawna.

BŁĄD MQRC_CFIL_ERROR
(2236, X'8BC') Niepoprawna struktura parametru listy całkowitej PCF lub struktura parametru listy całkowitej PCIF*64 .

BŁĄD MQRC_CFIN_
(2237, X'8BD') Niepoprawna struktura parametru liczby całkowitej PCF lub struktura parametru liczby całkowitej PCIF*64 .

BŁĄD MQRC_CFSF_ERROR
(2415, X'96F') Struktura parametru filtru łańcucha PCF w danych komunikatu jest niepoprawna.

BŁĄD MQRC_CFSL_ERROR
(2238, X'8BE') Niepoprawna struktura parametru listy łańcuchów PCF.

MQRC_CFST_BŁĄD
(2239, X'8BF') Niepoprawna struktura parametru łańcuchowego PCF.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

BŁĄD MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

BŁĄD MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Rozstrzyganie nazwy klastra nie powiodło się.

BŁĄD ZASOBU MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Błąd zasobu klastra.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') Opcja raportu COD nie jest poprawna dla kolejki XCF.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CONTENT_ERROR
2554 (X'09FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat powinien zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

MQRC_CONTEXT_HANDLE_ERROR,
(2097, X'831 ') Uchwyt kolejki, do którego istnieje odwołanie, nie zapisuje kontekstu.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Kontekst nie jest dostępny dla uchwytu kolejki, do którego się odwołuje.

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Niepoprawny parametr długości danych.

BŁĄD MQRC_DH_ERROR
(2135, X'857 ') Niepoprawna struktura nagłówka dystrybucji.

BŁĄD MQRC_DLH_ERROR
(2141, X'85D') Niepoprawna struktura nagłówka niedostarczonego komunikatu.

BŁĄD MQRC_EPH_ERROR
(2420, X' 974 ') Wbudowana struktura PCF jest niepoprawna.

BŁĄD UTRATY ważności MQRC_EXPIRY_ERROR
(2013, X'7DD') Niepoprawny czas ważności.

BŁĄD MQRC_FEEDBACK_ERROR
(2014, X'7DE') Kod zapisu czynności jest niepoprawny.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Globalny konflikt jednostek pracy.

BŁĄD MQRC_GROUP_ID_ERROR
(2258, X'8D2') Niepoprawny identyfikator grupy.

MQRC_HANDLE_IN_USE_FOR_UOW,
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD STERTY MQRC_HEADER_ERROR
(2142, X'85E') Niepoprawna struktura nagłówka MQ .

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

BŁĄD MQRC_IIH_ERROR
(2148, X'864 ') Niepoprawna struktura nagłówka informacji IMS .

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

MQRC_INCONSISTENT_PERSISTENCE
(2185, X'889 ') Niespójna specyfikacja trwałości.

MQRC_INCONSISTENT_UOW,
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

MQRC_LOCAL_UOW_CONFLICT (konflikt uow_rejestru)
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

MQRC_MD_BŁĄD
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

BŁĄD MQRC_MDE_ERROR
(2248, X'8C8') Niepoprawne rozszerzenie deskryptora komunikatu.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Brak kolejki odpowiedzi lub użyto MQPMO_SUPPRESS_REPLYTO

MQRC_MISSING_WIH (brak łącznika MQRC)
(2332, X'91C') Dane komunikatu nie rozpoczynają się od MQWIH.

BŁĄD MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Niepoprawne flagi komunikatu.

BŁĄD MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Długość komunikatu przekracza maksimum dla kolejki.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

BŁĄD MQRC_MSG_TYPE_ERROR
(2029, X'7ED') Niepoprawny typ komunikatu w deskryptorze komunikatu.

MQRC_MULTIPLE_POWODY
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') Brak dostępnych kolejek docelowych.

MQRC_NOT_OPEN_FOR_OUTPUT
(2039, X'7F7') Kolejka nie jest otwarta do wyprowadzania.

MQRC_NOT_OPEN_FOR_PASS_ALL-WSZYSTKIE
(2093, X'82D') Kolejka nie jest otwarta dla przekazywania całego kontekstu.

MQRC_NOT_OPEN_FOR_PASS_IDENT (identyfikator PASS)
(2094, X'82E') Kolejka nie jest otwarta dla przekazywania kontekstu tożsamości.

MQRC_NOT_OPEN_FOR_SET_ALL
(2095, X'82F') Kolejka nie jest otwarta dla ustawiania całego kontekstu.

MQRC_NOT_OPEN_FOR_SET_IDENT
(2096, X'830 ') Kolejka nie jest otwarta dla ustawionego kontekstu tożsamości.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

BŁĄD MQRC_OFFSET_
(2251, X'8CB') Przesunięcie segmentu komunikatu jest niepoprawne.

MQRC_OPEN_FAILED,
(2137, X'859 ') Obiekt nie został pomyślnie otwarty.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Niepoprawna oryginalna długość.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_PAGESET_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

BŁĄD MQRC_PCF_ERROR
(2149, X'865 ') Niepoprawne struktury PCF.

BŁĄD MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Trwałość jest niepoprawna.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') Kolejka nie obsługuje komunikatów trwałych.

BŁĄD MQRC_PMO_ERROR
(2173, X'87D') Niepoprawna struktura opcji umieszczania komunikatu.

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Niepoprawne flagi rekordu komunikatu.

BŁĄD MQRC_PRIORITY_ERROR
(2050, X'802 ') Priorytet komunikatu jest niepoprawny.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') Publikacja nie została dostarczona do żadnego z subskrybentów.

MQRC_PUT_INHIBITED
(2051, X'803 ') Wywołania umieszczania zablokowane dla kolejki, dla kolejki, do której ta kolejka jest rozstrzygana, lub dla tematu.

BŁĄD REKORDÓW MQRC_PUT_MSG_RECORDS
(2159, X'86F') Niepoprawne rekordy umieszczonych komunikatów.

MQRC_PUT_NOT_ZACHOWANE
(2479, X'09AF') Nie można zachować publikacji

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_FULL
(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') Brak miejsca na dysku dla kolejki.

MQRC_RECONNECT_FAILED
(2548, X'9F4') Po ponownym nawiązaniu połączenia wystąpił błąd podczas przywracania uchwytów dla połączenia z możliwością ponownego połączenia.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Niepoprawna liczba rekordów.

BŁĄD MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Opcje raportu w deskryptorze komunikatu są niepoprawne.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Rekordy odpowiedzi są niepoprawne.

BŁĄD MQRC_RFH_ERROR
(2334, X'91E') Niepoprawna struktura MQRFH lub MQRFH2 .

BŁĄD MQRC_RMH_ERROR
(2220, X'8AC') Niepoprawna struktura nagłówka komunikatu odniesienia.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') Długość danych w segmencie komunikatu wynosi zero.

MQRC_SEGMENTS_NOT_SUPPORTED (nieobsługiwane)
(2365, X'93D') Segmenty nie są obsługiwane.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Istnieje możliwy subskrybent publikacji, ale menedżer kolejek nie może sprawdzić, czy publikacja ma zostać wysłana do subskrybenta.

MQRC_STOPPED_BY_CLUSTER_EXIT,
(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

BŁĄD KLASY MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') Błąd klasy pamięci.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

MQRC_SYNCPOINT_NIEDOSTĘPNE
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

BŁĄD MQRC_TM_ERROR
(2265, X'8D9') Niepoprawna struktura komunikatu wyzwalacza.

BŁĄD MQRC_TMC_ERROR
(2191, X'88F') Niepoprawna struktura komunikatu wyzwalacza znaków.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

MQRC_UOW_NIEDOSTĘPNE
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

BŁĄD MQRC_WIH_ERROR
(2333, X'91D') Niepoprawna struktura MQWIH.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

BŁĄD MQRC_XQH_ERROR
(2260, X'8D4') Niepoprawna struktura nagłówka kolejki transmisji.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Uwagi dotyczące użycia tematu

1. Do korzystania z tematów mają zastosowanie następujące uwagi:

   a. W przypadku używania MQPUT do publikowania komunikatów w temacie, w którym jeden lub więcej subskrybentów tego tematu nie może otrzymać publikacji z powodu problemu z ich kolejką subskrybentów (na przykład jest pełna), kod przyczyny zwracany do wywołania MQPUT i zachowanie dostarczania jest zależne od ustawienia atrybutów PMSGDLV lub NPMSGDLV w temacie TOPIC. Jeśli określono opcję MQRO_DEAD_LETTER_Q, dostarczanie publikacji do kolejki niedostarczonych komunikatów lub odrzucanie komunikatu, gdy określono opcję MQRO_DISCARD_MSG, jest traktowane jako pomyślne dostarczenie komunikatu. Jeśli żadna z publikacji nie zostanie dostarczona, komenda MQPUT zwróci wartość MQRC_PUBLICATION_FAILURE. Może to wystąpić w następujących przypadkach:

   - Komunikat jest publikowany w temacie TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALL, a każda subskrypcja (trwała lub nie) ma kolejkę, która nie może odebrać publikacji.
   - Komunikat jest publikowany w temacie TOPIC z PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALLDUR, a trwała subskrypcja ma kolejkę, która nie może odebrać publikacji.

   Komenda MQPUT może zwrócić wartość MQRC_NONE, nawet jeśli w następujących przypadkach nie można było dostarczyć publikacji do niektórych subskrybentów:

   - Komunikat jest publikowany w temacie TOPIC z PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALLAVAIL, a każda subskrypcja, trwała lub nie, ma kolejkę, która nie może odebrać publikacji.
   - Komunikat jest publikowany w temacie TOPIC z PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALLDUR, a nietrwała subskrypcja ma kolejkę, która nie może odebrać publikacji.

   Za pomocą atrybutu tematu USEDLQ można określić, czy kolejka niedostarczonych komunikatów jest używana, gdy nie można dostarczyć komunikatów publikowania do poprawnej kolejki subskrybenta. Więcej informacji na temat używania parametru USEDLQ zawiera sekcja DEFINE TOPIC.

   b. Jeśli nie ma subskrybentów używanego tematu, opublikowany komunikat nie jest wysyłany do żadnej kolejki i jest odrzucany. Nie ma znaczenia, czy komunikat jest trwały, czy nietrwały, czy też ma nieograniczony czas utraty ważności lub ma czas utraty ważności. Jeśli nie ma subskrybentów, komunikat jest nadal usuwany. Wyjątkiem jest sytuacja, w której komunikat ma zostać zachowany. W takim przypadku, mimo że komunikat nie jest wysyłany do kolejek subskrybentów, jest on zapisywany w temacie, który ma zostać dostarczony do nowych subskrypcji lub do subskrybentów, którzy proszą o zachowane publikacje przy użyciu komendy MQSUBRQ.

MQPUT i MQPUT1

Do umieszczania komunikatów w kolejce można używać zarówno wywołań MQPUT, jak i MQPUT1 . Wywołanie, które ma być używane, zależy od okoliczności.

- Wywołanie MQPUT służy do umieszczania wielu komunikatów w tej samej kolejce.

  Najpierw wysyłane jest wywołanie MQOPEN określające opcję MQOO_OUTPUT, a następnie jedno lub więcej żądań MQPUT w celu dodania komunikatów do kolejki. Na końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1 .
- Wywołanie MQPUT1 służy do umieszczania w kolejce tylko jednego komunikatu.

  To wywołanie hermetyzuje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wykonane.

Kolejki docelowe

Poniższe uwagi dotyczą użycia kolejek docelowych:

1. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez użycia grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są szczegółowe warunki. Niektóre warunki dotyczą zarówno lokalnej, jak i zdalnej kolejki docelowej; inne dotyczą tylko zdalnych kolejek docelowych.

   Warunki, które mają zastosowanie do kolejek lokalnych i zdalnych miejsc docelowych

   - Wszystkie wywołania MQPUT znajdują się w tej samej jednostce pracy lub żadne z nich nie znajduje się w jednostce pracy.

     Należy pamiętać, że jeśli komunikaty są umieszczane w określonej kolejce w ramach pojedynczej jednostki pracy, komunikaty z innych aplikacji mogą być przeplatane sekwencją komunikatów w kolejce.
   - Wszystkie wywołania MQPUT są wykonywane przy użyciu tego samego uchwytu obiektu *Hobj*.

     W niektórych środowiskach sekwencja komunikatów jest również zachowywana, gdy używane są różne uchwyty obiektów, jeśli wywołania są wykonywane z tej samej aplikacji. Znaczenie *tej samej aplikacji* jest określane przez środowisko:

     – z/OS W systemie z/OSaplikacja jest następująca:

       - W przypadku systemu CICSzadanie CICS
       - W przypadku systemu IMSzadanie
       - W przypadku zadania wsadowego z/OS :

     – IBM i W systemie IBM iaplikacją jest zadanie.

     – ALW W systemie AIX, Linux, and Windowsaplikacja jest wątkiem.
   - Wszystkie komunikaty mają ten sam priorytet.
   - Komunikaty nie są umieszczane w kolejce klastra z określoną wartością MQOO_BIND_NOT_FIXED (lub z wartością MQOO_BIND_AS_Q_DEF, gdy atrybut kolejki DefBind ma wartość MQBND_BIND_NOT_FIXED).

   Dodatkowe warunki dotyczące zdalnych kolejek docelowych

   - Istnieje tylko jedna ścieżka od wysyłającego menedżera kolejek do docelowego menedżera kolejek.

     Jeśli niektóre komunikaty w sekwencji mogą znajdować się w innej ścieżce (na przykład ze względu na rekonfigurację, równoważenie ruchu lub wybór ścieżki na podstawie wielkości komunikatu), kolejność komunikatów w menedżerze kolejek docelowych nie może być zagwarantowana.
   - Komunikaty nie są tymczasowo umieszczane w kolejkach niedostarczonych komunikatów w nadawczych, pośrednich lub docelowych menedżerach kolejek.

Jeśli co najmniej jeden komunikat zostanie tymczasowo umieszczony w kolejce niedostarczonych komunikatów (na przykład z powodu tymczasowego zapełnienia kolejki transmisji lub kolejki docelowej), komunikaty mogą dotrzeć do kolejki docelowej poza kolejnością.

- Wszystkie komunikaty są trwałe lub nietrwałe.

  Jeśli kanał w trasie między menedżerami kolejek wysyłającym i docelowym ma ustawiony atrybut **NonPersistentMsgSpeed** na wartość MQNPMS_FAST, komunikaty nietrwałe mogą wyprzedzać komunikaty trwałe, co powoduje, że kolejność komunikatów trwałych względem komunikatów nietrwałych nie jest zachowywana. Zachowywana jest jednak kolejność komunikatów trwałych względem siebie i komunikatów nietrwałych względem siebie.

Jeśli te warunki nie są spełnione, można użyć grup komunikatów w celu zachowania kolejności komunikatów, ale wymaga to użycia obsługi grupowania komunikatów zarówno przez aplikacje wysyłające, jak i odbierające. Więcej informacji na temat grup komunikatów zawiera sekcja:

- MQMD-pole MsgFlags
- MQPMO_LOGICAL_ORDER
- MQGMO_LOGICAL_ORDER (MQGMO_LOGICAL_ORDER)

Listy dystrybucyjne

Poniższe uwagi dotyczą użycia list dystrybucyjnych.

Listy dystrybucyjne są obsługiwane w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

1. Komunikaty można umieszczać na liście dystrybucyjnej za pomocą programu MQPMO w wersji version-1 lub version-2 . Jeśli używany jest program MQPMO w wersji version-1 (lub program MQPMO w wersji version-2 z wartością `RecsPresent` równą zero), aplikacja nie może udostępniać żadnych rekordów komunikatów umieszczania ani rekordów odpowiedzi. Nie można zidentyfikować kolejek, w których wystąpiły błędy, jeśli komunikat został pomyślnie wysłany do niektórych kolejek na liście dystrybucyjnej, a nie do innych kolejek.

   Jeśli aplikacja udostępnia rekordy komunikatów umieszczania lub rekordy odpowiedzi, w polu `Version` należy ustawić wartość MQPMO_VERSION_2.

   Do wysyłania komunikatów do pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej, można również użyć programu MQPMO w wersji version-2 , upewniając się, że parametr `RecsPresent` ma wartość zero.
2. Parametry kodu zakończenia i kodu przyczyny są ustawione w następujący sposób:
   - Jeśli wszystkie operacje umieszczania w kolejkach na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, kod zakończenia i parametry kodu przyczyny są ustawiane w celu opisania wspólnego wyniku. W tym przypadku rekordy odpowiedzi MQRR (jeśli są udostępniane przez aplikację) nie są ustawione.

     Na przykład, jeśli każde umieszczenie powiedzie się, kod zakończenia i kod przyczyny zostaną ustawione na MQCC_OK i MQRC_NONE; jeśli każde umieszczenie nie powiedzie się, ponieważ wszystkie kolejki są zablokowane dla operacji umieszczania, parametry zostaną ustawione na MQCC_FAILED i MQRC_PUT_INHIBITED.
   - Jeśli operacje umieszczania w kolejkach na liście dystrybucyjnej nie powiodą się lub zakończą się niepowodzeniem w ten sam sposób:

- Parametr kodu zakończenia jest ustawiany na wartość MQCC_WARNING, jeśli co najmniej jedno umieszczenie powiodło się, lub na wartość MQCC_FAILED, jeśli wszystkie nie powiodły się.
- Parametr kodu przyczyny jest ustawiony na wartość MQRC_MULTIPLE_REASON.
- Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.

Jeśli operacja umieszczania w miejscu docelowym nie powiedzie się z powodu niepowodzenia operacji otwierania dla tego miejsca docelowego, pola w rekordzie odpowiedzi są ustawiane na wartości MQCC_FAILED i MQRC_OPEN_FAILED; to miejsce docelowe jest uwzględniane w pliku `InvalidDestCount`.

3. Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę lokalną, komunikat jest umieszczany w tej kolejce w normalnej formie (nie jako komunikat listy dystrybucyjnej). Jeśli więcej niż jedno miejsce docelowe jest tłumaczone na tę samą kolejkę lokalną, jeden komunikat jest umieszczany w kolejce dla każdego takiego miejsca docelowego.

   Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę zdalną, komunikat jest umieszczany w odpowiedniej kolejce transmisji. W przypadku gdy kilka miejsc docelowych jest rozstrzyganych do tej samej kolejki transmisji, pojedynczy komunikat listy dystrybucyjnej zawierający te miejsca docelowe może zostać umieszczony w kolejce transmisji, nawet jeśli te miejsca docelowe nie znajdowały się obok siebie na liście miejsc docelowych udostępnianej przez aplikację. Można to jednak zrobić tylko wtedy, gdy kolejka transmisji obsługuje komunikaty listy dystrybucyjnej (patrz sekcja DistLists).

   Jeśli kolejka transmisji nie obsługuje list dystrybucyjnych, jedna kopia komunikatu w normalnej formie jest umieszczana w kolejce transmisji dla każdego miejsca docelowego używającego tej kolejki transmisji.

   Jeśli lista dystrybucyjna z danymi komunikatu aplikacji jest zbyt duża dla kolejki transmisji, komunikat listy dystrybucyjnej jest dzielony na mniejsze komunikaty listy dystrybucyjnej, z których każdy zawiera mniej miejsc docelowych. Jeśli dane komunikatu aplikacji mieszczą się tylko w kolejce, nie można w ogóle używać komunikatów listy dystrybucyjnej, a menedżer kolejek generuje jedną kopię komunikatu w normalnej formie dla każdego miejsca docelowego używającego tej kolejki transmisji.

   Jeśli różne miejsca docelowe mają inny priorytet lub trwałość komunikatu (może to wystąpić, gdy aplikacja określa opcję MQPRI_PRIORITY_AS_Q_DEF lub MQPER_PERSISTENCE_AS_Q_DEF), komunikaty nie są przechowywane w tym samym komunikacie listy dystrybucji. Zamiast tego menedżer kolejek generuje tyle komunikatów listy dystrybucyjnej, ile jest potrzebnych do obsługi różnych wartości priorytetu i trwałości.

4. Umieszczenie na liście dystrybucyjnej może spowodować:
   - Pojedynczy komunikat listy dystrybucyjnej lub
   - Liczba mniejszych komunikatów z listy dystrybucyjnej lub
   - Połączenie komunikatów listy dystrybucyjnej i komunikatów normalnych, lub
   - Tylko normalne komunikaty.

   To, które z powyższych zdarzeń ma miejsce, zależy od tego, czy:
   - Miejsca docelowe na liście są lokalne, zdalne lub mieszane.
   - Miejsca docelowe mają taki sam priorytet i trwałość komunikatu.
   - Kolejki transmisji mogą przechowywać komunikaty listy dystrybucyjnej.
   - Maksymalna długość komunikatów kolejki transmisji jest wystarczająco duża, aby pomieścić komunikat w formie listy dystrybucyjnej.

   Jednak niezależnie od tego, która z powyższych sytuacji ma miejsce, każdy komunikat *fizyczny* będący wynikiem (to znaczy każdy normalny komunikat lub komunikat z listy dystrybucyjnej wynikający z operacji umieszczania) jest liczony jako *jeden* komunikat, gdy:
   - Sprawdzanie, czy aplikacja przekroczyła maksymalną dozwoloną liczbę komunikatów w jednostce pracy (patrz atrybut menedżera kolejek systemu **MaxUncommittedMsgs**).

- Sprawdzanie, czy warunki wyzwalania są spełnione.
- Zwiększanie głębokości kolejki i sprawdzanie, czy zostanie przekroczone maksymalne zapełnienie kolejki.

5. Każda zmiana definicji kolejek, która spowodowałaby, że uchwyt stałby się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana w ścieżce rozstrzygania), nie powoduje, że uchwyt listy dystrybucyjnej stałby się niepoprawny. Jednak powoduje to niepowodzenie dla tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnym wywołaniu MQPUT.

Nagłówki

Jeśli komunikat jest umieszczany z jedną lub większą liczbą struktur nagłówka IBM MQ na początku danych komunikatu aplikacji, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka, aby sprawdzić, czy są one poprawne. Jeśli menedżer kolejek wykryje błąd, wywołanie zakończy się niepowodzeniem z odpowiednim kodem przyczyny. Przeprowadzone kontrole różnią się w zależności od konkretnych struktur, które są obecne:

- Sprawdzenia są wykonywane tylko wtedy, gdy w wywołaniu MQPUT lub MQPUT1 używana jest komenda MQMD w wersji version-2 lub nowszej. Sprawdzanie nie jest wykonywane w przypadku użycia deskryptora MQMD version-1 , nawet jeśli na początku danych komunikatu znajduje się deskryptor MQMDE.
- Nie jest sprawdzana poprawność struktur, które nie są obsługiwane przez menedżer kolejek lokalnych, oraz struktur następujących po pierwszym MQDLH w komunikacie.
- Poprawność struktur MQDH i MQMDE jest całkowicie sprawdzana przez menedżer kolejek.
- Poprawność innych struktur jest sprawdzana częściowo przez menedżer kolejek (nie wszystkie pola są sprawdzane).

Ogólne sprawdzenia wykonywane przez menedżera kolejek obejmują:

- Pole `StrucId` musi być poprawne.
- Pole `Version` musi być poprawne.
- Pole `StrucLength` musi określać wartość, która jest wystarczająco duża, aby uwzględnić strukturę oraz dane o zmiennej długości tworzące część struktury.
- Pole `CodedCharSetId` nie może mieć wartości zero lub wartość ujemna, która nie jest poprawna (MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR i MQCCSI_UNDEFINED nie są poprawne w większości struktur nagłówków IBM MQ).
- Parametr **`BufferLength`** wywołania musi określać wartość, która jest wystarczająco duża, aby uwzględnić strukturę (struktura nie może wykraczać poza koniec komunikatu).

Oprócz ogólnych kontroli konstrukcji muszą być spełnione następujące warunki:

- Suma długości struktur w komunikacie PCF musi być równa długości określonej przez parametr **`BufferLength`** w wywołaniu MQPUT lub MQPUT1 . Komunikat PCF to komunikat o nazwie formatu MQFMT_ADMIN, MQFMT_EVENT lub MQFMT_PCF.
- Struktura IBM MQ nie może zostać obcięta, z wyjątkiem następujących sytuacji, w których dozwolone są obcięte struktury:
  - Komunikaty, które są komunikatami raportów.
  - Komunikaty PCF.
  - Komunikaty zawierające strukturę MQDLH. (Struktury następujące po pierwszym MQDLH mogą zostać obcięte; struktury poprzedzające MQDLH nie mogą.)
- Struktura IBM MQ nie może być podzielona na dwa lub więcej segmentów; struktura musi być całkowicie zawarta w jednym segmencie.

Buforuj

W przypadku języka programowania Visual Basic mają zastosowanie następujące punkty:

- Jeśli wielkość parametru **Buffer** jest mniejsza niż długość określona przez parametr **BufferLength** , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_BUFFER_LENGTH_ERROR.
- Parametr **Buffer** jest zadeklarowany jako typu `String`. Jeśli dane, które mają zostać umieszczone w kolejce, nie są typu `String`, należy użyćWywołanie MQPUTAny zamiast wywołania MQPUT.

  Wywołanie MQPUTAny ma takie same parametry jak wywołanie MQPUT, z tym wyjątkiem, że parametr **Buffer** jest zadeklarowany jako typ Any, co umożliwia umieszczenie w kolejce dowolnego typu danych. Oznacza to jednak, że nie można sprawdzić, czy wielkość pliku `Buffer` wynosi co najmniej `BufferLength` bajtów.

Wywołanie C

```
MQPUT (Hconn, Hobj, &MsgDesc, &PutMsgOpts, BufferLength, Buffer,
      &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
     CALL 'MQPUT' USING HCONN, HOBJ, MSGDESC, PUTMSGOPTS, BUFFERLENGTH,
                        BUFFER, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQPUT (Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer,
           CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQPUT,(HCONN,HOBJ,MSGDESC,PUTMSGOPTS,BUFFERLENGTH, X
               BUFFER,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

Windows

```
MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, BufferLength, Buffer, CompCode,
      Reason
```

Zadeklaruj parametry w następujący sposób:

```
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 -Umieść jeden komunikat

Wywołanie MQPUT1 umieszcza jeden komunikat w kolejce, na liście dystrybucyjnej lub w temacie.

Kolejka, lista dystrybucyjna lub temat nie muszą być otwarte.

Składnia

MQPUT1 (*Hconn*, *ObjDesc*, *MsgDesc*, *PutMsgOpts*, *BufferLength*, *Buffer*, *CompCode*, *Przyczyna*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

z/OS W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

ObjDesc
Typ: MQOD-wejście/wyjście

Jest to struktura identyfikująca kolejkę, do której komunikat jest dodawany, lub temat, do którego komunikat jest publikowany. Szczegółowe informacje można znaleźć w sekcji “MQOD-deskryptor obiektu” na stronie 493.

Jeśli struktura jest kolejką, użytkownik musi mieć uprawnienia do otwarcia kolejki dla danych wyjściowych. Kolejka nie może być kolejką modelową.

MsgDesc
Typ: MQMD-wejście/wyjście

Ta struktura opisuje atrybuty wysyłanego komunikatu i odbiera informacje zwrotne po zakończeniu żądania umieszczenia. Szczegółowe informacje można znaleźć w sekcji “MQMD-deskryptor komunikatu” na stronie 432.

Jeśli aplikacja udostępnia strukturę MQMD version-1 , dane komunikatu można poprzedzić strukturą MQMDE, aby określić wartości pól, które istnieją w strukturze MQMD version-2 , ale nie w strukturze version-1. Ustaw wartość pola `Format` w polu MQMD na MQFMT_MD_EXTENSION, aby wskazać, że istnieje MQMDE. Więcej informacji na temat zawiera sekcja “MQMDE-rozszerzenie deskryptora komunikatu” na stronie 485.

Aplikacja nie musi udostępniać struktury MQMD, jeśli w polu `MsgHandle` struktury MQGMO lub w polach `OriginalMsgHandle` lub `NewMsgHandle` struktury MQPMO podano poprawny uchwyt komunikatu. Jeśli w jednym z tych pól nie zostanie podana żadna wartość, deskryptor komunikatu jest pobierany z deskryptora powiązanego z uchwytami komunikatów.

PutMsg-opcje
Typ: MQPMO-wejście/wyjście

Szczegółowe informacje można znaleźć w sekcji “MQPMO-opcje umieszczania komunikatów” na stronie 514.

BufferLength
Typ: MQLONG-input

Długość komunikatu w pliku `Buffer`. Zero jest poprawne i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit zależy od różnych czynników; opis parametru **`BufferLength`** można znaleźć w sekcji “MQPUT-komunikat umieszczania” na stronie 774 .

Buforuj
Typ: MQBYTExBuffer-długość-wejście

Jest to bufor zawierający dane komunikatu aplikacji, które mają zostać wysłane. Wyrównaj bufor na granicy odpowiedniej do rodzaju danych w komunikacie. Wyrównanie 4-bajtowe jest odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówków IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli plik `Buffer` zawiera dane znakowe lub liczbowe, należy ustawić pola `CodedCharSetId` i `Encoding` w parametrze **`MsgDesc`** na wartości odpowiednie dla danych. Umożliwia to odbiorcy komunikatu przekształcanie danych (w razie potrzeby) na zestaw znaków i kodowanie używane przez odbiorcę.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT1 muszą mieć zestaw znaków i kodowanie lokalnego menedżera kolejek (określone przez atrybut menedżera kolejek systemu **`CodedCharSetId`** i parametr MQENC_NATIVE).

W języku programowania C parametr jest zadeklarowany jako wskaźnik do unieważnienia; jako parametr można podać adres dowolnego typu danych.

Jeśli parametr **BufferLength** ma wartość zero, nie jest przywoływany parametr Buffer ; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość null.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako CompCode.

Jeśli CompCode ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli CompCode ma wartość MQCC_WARNING:

MQRC_MULTIPLE_POWODY
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

MQRC_INCOMPLETE_GROUP
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

MQRC_INCOMPLETE_MSG
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

MQRC_PRIORITY_PRZEKROCZONE_MAKSIMUM
(2049, X'801 ') Priorytet komunikatu przekracza maksymalną obsługiwaną wartość.

MQRC_UNKNOWN_REPORT_OPCJA
(2104, X'838 ') Opcje raportu w deskryptorze komunikatu nie zostały rozpoznane.

Jeśli CompCode ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_ALIAS_BASE_Q_TYPE_ERROR
(2001, X'7D1') Kolejka podstawowa aliasu nie jest poprawnym typem.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_BACKED_OUT
(2003, X'7D3') Jednostka pracy wycofała się.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'7D4') Niepoprawny parametr buforu.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_NIEDOSTĘPNE
(2345, X' 929 ') narzędzie sprzęgające niedostępne.

MQRC_CF_STRUC_AUTH_FAILED
(2348, X'92C') Sprawdzanie autoryzacji struktury narzędzia CF nie powiodło się.

MQRC_CF_STRUC_BŁĄD
(2349, X'92D') Niepoprawna struktura narzędzia CF.

MQRC_CF_STRUC_FAILED
(2373, X' 945 ') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Nagłówek listy struktury narzędzia CF w użyciu.

BŁĄD MQRC_CFGR_ERROR
(2416, X' 970 ') Struktura parametru grupy PCF MQCFGR w danych komunikatu jest niepoprawna.

BŁĄD MQRC_CFH_ERROR
(2235, X'8BB') Niepoprawna struktura nagłówka PCF.

MQRC_CFIF_BŁĄD
(2414, X'96E') Struktura parametru filtru liczby całkowitej PCF w danych komunikatu jest niepoprawna.

BŁĄD MQRC_CFIL_ERROR
(2236, X'8BC') Niepoprawna struktura parametru listy całkowitej PCF lub struktura parametru listy całkowitej PCIF*64 .

BŁĄD MQRC_CFIN_
(2237, X'8BD') Niepoprawna struktura parametru liczby całkowitej PCF lub struktura parametru liczby całkowitej PCIF*64 .

BŁĄD MQRC_CFSF_ERROR
(2415, X'96F') Struktura parametru filtru łańcucha PCF w danych komunikatu jest niepoprawna.

BŁĄD MQRC_CFSL_ERROR
(2238, X'8BE') Niepoprawna struktura parametru listy łańcuchów PCF.

MQRC_CFST_BŁĄD
(2239, X'8BF') Niepoprawna struktura parametru łańcuchowego PCF.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

BŁĄD MQRC_CLUSTER_EXIT_ERROR
(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

BŁĄD MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Rozstrzyganie nazwy klastra nie powiodło się.

BŁĄD ZASOBU MQRC_CLUSTER_RESOURCE_ERROR
(2269, X'8DD') Błąd zasobu klastra.

MQRC_COD_NOT_VALID_FOR_XCF_Q
(2106, X'83A') Opcja raportu COD nie jest poprawna dla kolejki XCF.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

MQRC_CONNECTION_QUIESCING,
(2202, X'89A') wygaszanie połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

BŁĄD MQRC_CONTENT_ERROR
2554 (X'09FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat może zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

MQRC_CONTEXT_HANDLE_ERROR,
(2097, X'831 ') Uchwyt kolejki, do którego istnieje odwołanie, nie zapisuje kontekstu.

MQRC_CONTEXT_NOT_AVAILABLE
(2098, X'832 ') Kontekst nie jest dostępny dla uchwytu kolejki, do którego się odwołuje.

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Niepoprawny parametr długości danych.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

BŁĄD MQRC_DEF_XMIT_Q_TYPE_ERROR
(2198, X'896 ') Domyślna kolejka transmisji nie jest lokalna.

MQRC_DEF_XMIT_Q_USAGE_ERROR,
(2199, X'897 ') Błąd użycia domyślnej kolejki transmisji.

BŁĄD MQRC_DH_ERROR
(2135, X'857 ') Niepoprawna struktura nagłówka dystrybucji.

BŁĄD MQRC_DLH_ERROR
(2141, X'85D') Niepoprawna struktura nagłówka niedostarczonego komunikatu.

BŁĄD MQRC_EPH_ERROR
(2420, X' 974 ') Wbudowana struktura PCF jest niepoprawna.

BŁĄD UTRATY ważności MQRC_EXPIRY_ERROR
(2013, X'7DD') Niepoprawny czas ważności.

BŁĄD MQRC_FEEDBACK_ERROR
(2014, X'7DE') Kod zapisu czynności jest niepoprawny.

MQRC_GLOBAL_UOW_CONFLICT
(2351, X'92F') Globalny konflikt jednostek pracy.

BŁĄD MQRC_GROUP_ID_ERROR
(2258, X'8D2') Niepoprawny identyfikator grupy.

MQRC_HANDLE_IN_USE_FOR_UOW,
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

MQRC_HANDLE_NOT_AVAILABLE
(2017, X'7E1') Brak dostępnych uchwytów.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD STERTY MQRC_HEADER_ERROR
(2142, X'85E') IBM MQ struktura nagłówka jest niepoprawna.

BŁĄD MQRC_IIH_ERROR
(2148, X'864 ') Niepoprawna struktura nagłówka informacji IMS .

MQRC_LOCAL_UOW_CONFLICT (konflikt uow_rejestru)
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

MQRC_MD_BŁĄD
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

BŁĄD MQRC_MDE_ERROR
(2248, X'8C8') Niepoprawne rozszerzenie deskryptora komunikatu.

MQRC_MISSING_REPLY_TO_Q
(2027, X'7EB') Brak kolejki odpowiedzi.

MQRC_MISSING_WIH (brak łącznika MQRC)
(2332, X'91C') Dane komunikatu nie rozpoczynają się od MQWIH.

BŁĄD MQRC_MSG_FLAGS_ERROR
(2249, X'8C9') Niepoprawne flagi komunikatu.

BŁĄD MQRC_MSG_SEQ_NUMBER_ERROR
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

MQRC_MSG_TOO_BIG_FOR_Q
(2030, X'7EE') Długość komunikatu przekracza maksimum dla kolejki.

MQRC_MSG_TOO_BIG_FOR_Q_MGR
(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

BŁĄD MQRC_MSG_TYPE_ERROR
(2029, X'7ED') Niepoprawny typ komunikatu w deskryptorze komunikatu.

MQRC_MULTIPLE_POWODY
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

MQRC_NO_DESTINATIONS_AVAILABLE
(2270, X'8DE') Brak dostępnych kolejek docelowych.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

MQRC_OBJECT_IN_UŻYTK
(2042, X'7FA') Obiekt jest już otwarty z opcjami powodującymi konflikt.

MQRC_OBJECT_LEVEL_NIEKOMPATYBILNY
(2360, X' 938 ') Poziom obiektu nie jest zgodny.

BŁĄD MQRC_OBJECT_NAME_ERROR
(2152, X'868 ') Niepoprawna nazwa obiektu.

MQRC_OBJECT_NOT_UNIQUE (obiekt MQRC_NOT_UNIQUE)
(2343, X' 927 ') Obiekt nie jest unikalny.

Błąd MQRC_OBJECT_Q_MGR_NAME_ERROR
(2153, X'869 ') Niepoprawna nazwa menedżera kolejek obiektu.

BŁĄD REKORDÓW MQRC_OBJECT_RECORDS
(2155, X'86B') Rekordy obiektów są niepoprawne.

BŁĄD MQRC_OBJECT_TYPE_ERROR
(2043, X'7FB') Niepoprawny typ obiektu.

BŁĄD OD_MQRC
(2044, X'7FC') Niepoprawna struktura deskryptora obiektu.

BŁĄD MQRC_OFFSET_
(2251, X'8CB') Przesunięcie segmentu komunikatu jest niepoprawne.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_ORIGINAL_LENGTH_ERROR
(2252, X'8CC') Niepoprawna oryginalna długość.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_PAGESET_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

BŁĄD MQRC_PCF_ERROR
(2149, X'865 ') Niepoprawne struktury PCF.

BŁĄD MQRC_PERSISTENCE_ERROR
(2047, X'7FF') Trwałość jest niepoprawna.

MQRC_PERSISTENT_NOT_ALLOWED
(2048, X'800 ') Kolejka nie obsługuje komunikatów trwałych.

BŁĄD MQRC_PMO_ERROR
(2173, X'87D') Niepoprawna struktura opcji umieszczania komunikatu.

MQRC_PMO_RECORD_FLAGS_ERROR
(2158, X'86E') Niepoprawne flagi rekordu komunikatu.

BŁĄD MQRC_PRIORITY_ERROR
(2050, X'802 ') Priorytet komunikatu jest niepoprawny.

MQRC_PUBLICATION_FAILURE
(2502, X'9C6') Publikacja nie została dostarczona do żadnego z subskrybentów.

MQRC_PUT_INHIBITED
(2051, X'803 ') Wywołania umieszczania zablokowane dla kolejki.

BŁĄD REKORDÓW MQRC_PUT_MSG_RECORDS
(2159, X'86F') Niepoprawne rekordy umieszczonych komunikatów.

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

MQRC_Q_FULL
(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_QUIESCING,
(2161, X'871 ') wygaszanie menedżera kolejek.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_Q_SPACE_NOT_AVAILABLE
(2056, X'808 ') Brak miejsca na dysku dla kolejki.

BŁĄD MQRC_Q_TYPE_ERROR
(2057, X'809 ') Niepoprawny typ kolejki.

MQRC_RECS_PRESENT_ERROR
(2154, X'86A') Niepoprawna liczba rekordów.

BŁĄD NAZWY MQRC_REMOTE_Q_NAME_ERROR
(2184, X'888 ') Niepoprawna nazwa kolejki zdalnej.

BŁĄD MQRC_REPORT_OPTIONS_ERROR
(2061, X'80D') Opcje raportu w deskryptorze komunikatu są niepoprawne.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_RESPONSE_RECORDS_ERROR
(2156, X'86C') Rekordy odpowiedzi są niepoprawne.

BŁĄD MQRC_RFH_ERROR
(2334, X'91E') Niepoprawna struktura MQRFH lub MQRFH2 .

BŁĄD MQRC_RMH_ERROR
(2220, X'8AC') Niepoprawna struktura nagłówka komunikatu odniesienia.

BŁĄD MQRC_SECURITY_ERROR
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

MQRC_SEGMENT_LENGTH_ZERO
(2253, X'8CD') Długość danych w segmencie komunikatu wynosi zero.

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Istnieje możliwy subskrybent publikacji, ale menedżer kolejek nie może sprawdzić, czy publikacja ma zostać wysłana do subskrybenta.

MQRC_STOPPED_BY_CLUSTER_EXIT,
(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

BŁĄD KLASY MQRC_STORAGE_CLASS_ERROR
(2105, X'839 ') Błąd klasy pamięci.

MQRC_STORAGE_MEDIUM_FULL
(2192, X'890 ') Nośnik pamięci zewnętrznej jest pełny.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

MQRC_SYNCPOINT_LIMIT_REACHED
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

MQRC_SYNCPOINT_NIEDOSTĘPNE
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

BŁĄD MQRC_TM_ERROR
(2265, X'8D9') Niepoprawna struktura komunikatu wyzwalacza.

BŁĄD MQRC_TMC_ERROR
(2191, X'88F') Niepoprawna struktura komunikatu wyzwalacza znaków.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

MQRC_UNKNOWN_ALIAS_BASE_Q
(2082, X'822 ') Nieznana kolejka podstawowa aliasów.

MQRC_UNKNOWN_DEF_XMIT_Q
(2197, X'895 ') Nieznana domyślna kolejka transmisji.

MQRC_UNKNOWN_OBJECT_NAME
(2085, X'825 ') Nieznana nazwa obiektu.

MQRC_UNKNOWN_OBJECT_Q_MGR
(2086, X'826 ') Nieznany menedżer kolejek obiektów.

MQRC_UNKNOWN_REMOTE_Q_MGR
(2087, X'827 ') Nieznany zdalny menedżer kolejek.

MQRC_UNKNOWN_XMIT_Q
(2196, X'894 ') Nieznana kolejka transmisji.

MQRC_UOW_ENLISTMENT_ERROR
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

MQRC_UOW_MIX_NOT_SUPPORTED
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

MQRC_UOW_NIEDOSTĘPNE
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

BŁĄD MQRC_WIH_ERROR
(2333, X'91D') Niepoprawna struktura MQWIH.

MQRC_WRONG_CF_LEVEL
(2366, X'93E') Struktura narzędzia CF jest na niewłaściwym poziomie.

MQRC_WRONG_MD_VERSION
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

BŁĄD MQRC_XMIT_Q_TYPE_ERROR
(2091, X'82B') Kolejka transmisji nie jest lokalna.

MQRC_XMIT_Q_USAGE_ERROR,
(2092, X'82C') Kolejka transmisji z niewłaściwym użyciem.

BŁĄD MQRC_XQH_ERROR
(2260, X'8D4') Niepoprawna struktura nagłówka kolejki transmisji.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Zarówno wywołania MQPUT, jak i MQPUT1 mogą być używane do umieszczania komunikatów w kolejce. Wywołanie, które ma być używane, zależy od okoliczności:
   - Wywołanie MQPUT służy do umieszczania wielu komunikatów w *tej samej* kolejce.

     Najpierw wysyłane jest wywołanie MQOPEN określające opcję MQOO_OUTPUT, a następnie jedno lub więcej żądań MQPUT w celu dodania komunikatów do kolejki. Na końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1 .
   - Wywołanie MQPUT1 służy do umieszczania w kolejce tylko *jednego* komunikatu.

     To wywołanie hermetyzuje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wykonane.
2. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez użycia grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są pewne warunki. Jednak w większości środowisk wywołanie MQPUT1 nie spełnia tych warunków i nie zachowuje kolejności komunikatów. W tych środowiskach należy użyć wywołania MQPUT. Szczegółowe informacje na ten temat zawiera sekcja Uwagi dotyczące użycia MQPUT .
3. Do umieszczania komunikatów na listach dystrybucyjnych można użyć wywołania MQPUT1 . Ogólne informacje na ten temat zawierają uwagi dotyczące używania wywołań MQOPEN i MQPUT.

   Listy dystrybucyjne są obsługiwane w następujących środowiskach:
   - AIX
   - IBM i
   - Linux
   - Windows

   i dla klientów IBM MQ połączonych z tymi systemami.

   Podczas używania wywołania MQPUT1 występują następujące różnice:

   a. Jeśli aplikacja udostępnia rekordy odpowiedzi MQRR, muszą być one udostępniane przy użyciu struktury MQOD; nie mogą być udostępniane przy użyciu struktury MQPMO.

   b. Kod przyczyny MQRC_OPEN_FAILED nie jest nigdy zwracany przez funkcję MQPUT1 w rekordach odpowiedzi. Jeśli otwarcie kolejki nie powiedzie się, rekord odpowiedzi dla tej kolejki zawiera kod przyczyny będący wynikiem operacji otwierania.

      Jeśli operacja otwierania kolejki powiedzie się z kodem zakończenia MQCC_WARNING, kod zakończenia i kod przyczyny w rekordzie odpowiedzi dla tej kolejki są zastępowane przez kody zakończenia i przyczyny wynikające z operacji umieszczania.

      Podobnie jak w przypadku wywołań MQOPEN i MQPUT, menedżer kolejek ustawia rekordy odpowiedzi (jeśli zostały podane) tylko wtedy, gdy wynik wywołania nie jest taki sam dla wszystkich kolejek na liście dystrybucyjnej. Jest to wskazywane przez zakończenie wywołania z kodem przyczyny MQRC_MULTIPLE_REASON.
4. Jeśli wywołanie MQPUT1 jest używane do umieszczenia komunikatu w kolejce klastra, wywołanie zachowuje się tak, jakby w wywołaniu MQOPEN podano parametr MQOO_BIND_NOT_FIXED.
5. Jeśli komunikat jest umieszczany z jedną lub większą liczbą struktur nagłówka IBM MQ na początku danych komunikatu aplikacji, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka, aby

sprawdzić, czy są one poprawne. Więcej informacji na ten temat zawierają uwagi dotyczące użycia wywołania MQPUT.

6. Jeśli wystąpi więcej niż jedna sytuacja ostrzegawcza (patrz parametr **CompCode**), zwrócony kod przyczyny jest pierwszym z poniższej listy, który ma zastosowanie:
   a. MQRC_MULTIPLE_POWODY
   b. MQRC_INCOMPLETE_MSG
   c. MQRC_INCOMPLETE_GROUP
   d. MQRC_PRIORITY_PRZEKROCZONE_MAXIMUM lub MQRC_UNKNOWN_REPORT_OPTION
7. W przypadku języka programowania Visual Basic mają zastosowanie następujące punkty:
   - Jeśli wielkość parametru **Buffer** jest mniejsza niż długość określona przez parametr **BufferLength** , wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_BUFFER_LENGTH_ERROR.
   - Parametr **Buffer** jest zadeklarowany jako typu `String`. Jeśli dane, które mają zostać umieszczone w kolejce, nie są typu `String`, należy użyćMQPUT1Any zamiast MQPUT1.

     Wywołanie MQPUT1Any ma takie same parametry jak wywołanie MQPUT1 , z wyjątkiem tego, że parametr **Buffer** jest zadeklarowany jako typ `Any`, co umożliwia umieszczenie w kolejce dowolnego typu danych. Oznacza to jednak, że nie można sprawdzić, czy wielkość pliku `Buffer` wynosi co najmniej `BufferLength` bajtów.
8. Po wywołaniu wywołania MQPUT1 z opcją MQPMO_SYNCPOINT domyślne zachowanie zmienia się, tak że operacja umieszczania jest wykonywana asynchronicznie. Może to spowodować zmianę zachowania niektórych aplikacji, które opierają się na pewnych polach w zwracanych strukturach MQOD i MQMD, ale które teraz zawierają niezdefiniowane wartości. Aplikacja może określić parametr MQPMO_SYNC_RESPONSE, aby zapewnić, że operacja umieszczania jest wykonywana synchronicznie i że wszystkie odpowiednie wartości pól są zakończone.

Wywołanie C

```
MQPUT1 (Hconn, &ObjDesc, &MsgDesc, &PutMsgOpts,
       BufferLength, Buffer, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
     CALL 'MQPUT1' USING HCONN, OBJDESC, MSGDESC, PUTMSGOPTS,
                         BUFFERLENGTH, BUFFER, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQPUT1 (Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
            CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
         CALL MQPUT1,(HCONN,OBJDESC,MSGDESC,PUTMSGOPTS,BUFFERLENGTH,   X
               BUFFER,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

Windows

```
MQPUT1 Hconn, ObjDesc, MsgDesc, PutMsgOpts, BufferLength, Buffer,
       CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-ustawianie atrybutów obiektu

Wywołanie MQSET służy do zmiany atrybutów obiektu reprezentowanego przez uchwyt. Obiekt musi być kolejką.

Składnia

MQSET (*Hconn*, *Hobj*, *SelectorCount*, *Selectors*, *IntAttrCount*, *IntAttrs*, *CharAttrLength*, *CharAttrs*, *Compcode*, *Reason*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość Hconn została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

z/OS W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Hobj
Typ: MQHOBJ-input

Ten uchwyt reprezentuje obiekt kolejki z atrybutami, które mają zostać ustawione. Uchwyt został zwrócony przez poprzednie wywołanie MQOPEN, które określało opcję MQOO_SET.

SelectorCount
Typ: MQLONG-input

Jest to liczba selektorów, które są dostarczane w tablicy `Selectors` . Jest to liczba atrybutów, które mają zostać ustawione. Zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

Selektory
Typ: MQLONGxSelectorCount-input

Jest to tablica selektorów atrybutów **`SelectorCount`** . Każdy selektor identyfikuje atrybut (liczbę całkowitą lub znak) z wartością, która ma zostać ustawiona.

Każdy selektor musi być poprawny dla typu kolejki reprezentowanego przez `Hobj` . Dozwolone są tylko niektóre wartości MQIA_ * i MQCA_ *, wymienione poniżej.

Selektory można określić w dowolnej kolejności. Wartości atrybutów odpowiadające selektorom atrybutów całkowitych (MQIA_ * selectors) muszą być określone w pliku `IntAttrs` w tej samej kolejności, w jakiej występują one w produkcie `Selectors`. Wartości atrybutów odpowiadające selektorom atrybutów znakowych (MQCA_ * selectors) muszą być określone w pliku `CharAttrs` w tej samej kolejności, w jakiej występują te selektory. Selektory MQIA_ * mogą być przeplatane z selektorami MQCA_ *. Ważna jest tylko kolejność względna w obrębie każdego typu.

Ten sam selektor można określić więcej niż jeden raz. W takim przypadku zostanie zastosowana ostatnia wartość określona dla konkretnego selektora.

Uwaga:

1. Selektory atrybutów całkowitych i znakowych są przydzielane w dwóch różnych zakresach: selektory MQIA_ * rezydują w zakresie od MQIA_FIRST do MQIA_LAST, a selektory MQCA_ * w zakresie od MQCA_FIRST do MQCA_LAST.

   Dla każdego zakresu stałe MQIA_LAST_USED i MQCA_LAST_USED definiują najwyższą wartość akceptowaną przez menedżer kolejek.
2. Jeśli najpierw wystąpią wszystkie selektory MQIA_ *, można użyć tych samych numerów elementów do adresowania odpowiednich elementów w tablicach `Selectors` i `IntAttrs` .

3. Jeśli parametr **SelectorCount** ma wartość zero, parametr Selectors nie jest przywoływany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość null.

Atrybuty, które można ustawić, są wymienione w poniższej tabeli. Przy użyciu tego wywołania nie można ustawić żadnych innych atrybutów. W przypadku selektorów atrybutów MQCA_ * stała definiująca długość łańcucha (w bajtach), który jest wymagany w produkcie CharAttrs , jest podawana w nawiasach.

Tabela 554. Selektory atrybutów MQSET dla kolejek

Selektor	Opis	Uwaga
MQCA_TRIGGER_DATA	Dane wyzwalacza (MQ_TRIGGER_DATA_LENGTH).	
MQIA_DIST_LISTS	Obsługa listy dystrybucyjnej.	1
MQIA_INHIBIT_GET	Określa, czy operacje pobierania są dozwolone.	
MQIA_INHIBIT_PUT	Określa, czy operacje umieszczania (put) są dozwolone.	
MQIA_TRIGGER_CONTROL	Element sterujący wyzwalacza.	
MQIA_TRIGGER_DEPTH	Wyzwalacz uruchamiany zapełnieniem.	
MQIA_TRIGGER_MSG_PRIORITY,	Priorytet komunikatu progowego dla wyzwalaczy.	
MQIA_TRIGGER_TYPE	Typ wyzwalacza.	

Uwaga:

1. Obsługiwane tylko na następujących platformach:
   - AIX
   - IBM i
   - Linux
   - Windows

   i dla IBM MQ MQI clients połączonych z tymi systemami.

IntAttrLiczba
Typ: MQLONG-input

Jest to liczba elementów w tablicy IntAttrs i musi to być co najmniej liczba selektorów MQIA_ * w parametrze **Selectors** . Zero jest poprawną wartością, jeśli nie ma żadnej wartości.

IntAttrs
Typ: MQLONGxIntAttrCount -wejście

Jest to tablica wartości atrybutów całkowitoliczbowych IntAttrCount . Wartości tych atrybutów muszą być w tej samej kolejności, co selektory MQIA_ * w tablicy Selectors .

Jeśli parametr **IntAttrCount** lub **SelectorCount** ma wartość zero, IntAttrs nie jest przywoływany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 może mieć wartość null.

CharAttrDługość
Typ: MQLONG-input

Jest to długość w bajtach parametru **CharAttrs** , która musi być sumą długości atrybutów znakowych określonych w tablicy Selectors . Wartość zero jest poprawna, jeśli w produkcie Selectorsnie ma selektorów MQCA_ *.

CharAttrs
Typ: MQCHAR x CharAttrLength-input

Jest to bufor zawierający połączone ze sobą wartości atrybutów znakowych. Długość buforu jest określona przez parametr **CharAttrLength** .

Atrybuty znaków muszą być podane w tej samej kolejności, co selektory MQCA_ * w tablicy `Selectors` . Długość każdego atrybutu znakowego jest stała (patrz sekcja Selektory). Jeśli wartość, która ma być ustawiona dla atrybutu, zawiera mniej niepustych znaków niż zdefiniowana długość atrybutu, należy dopełnić wartość w polu `CharAttrs` do prawej strony odstępami, aby wartość atrybutu była zgodna ze zdefiniowaną długością atrybutu.

Jeśli parametr **CharAttrLength** lub **SelectorCount** ma wartość zero, `CharAttrs` nie jest przywoływany; w tym przypadku adres parametru przekazany przez programy napisane w języku C lub System/390 może mieć wartość null.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako `CompCode`.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli `CompCode` ma wartość MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście API nie powiodło się.

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

MQRC_CF_NIEDOSTĘPNE
(2345, X' 929 ') Obiekt sprzęgania nie jest dostępny.

MQRC_CF_STRUC_FAILED
(2373, X' 945 ') Struktura narzędzia CF nie powiodła się.

MQRC_CF_STRUC_IN_UŻYJ
(2346, X'92A') Struktura narzędzia CF w użyciu.

MQRC_CF_STRUC_LIST_HDR_IN_USE
(2347, X'92B') Nagłówek listy struktury narzędzia CF w użyciu.

BŁĄD MQRC_CHAR_ATTR_LENGTH_ERROR
(2006, X'7D6') Niepoprawna długość atrybutów znakowych.

BŁĄD MQRC_CHAR_ATTRS_ERROR
(2007, X'7D7') Niepoprawny łańcuch atrybutów znaku.

MQRC_CICS_WAIT_FAILED
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

MQRC_CONNECTION_NOT_AUTHORIZED (nieautoryzowany)
(2217, X'8A9') Brak uprawnień do połączenia.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

MQRC_DB2_NOT_AVAILABLE
(2342, X' 926 ') Podsystem Db2 nie jest dostępny.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_HOBJ_ERROR
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

BŁĄD MQRC_INHIBIT_VALUE_ERROR
(2020, X'7E4') Niepoprawna wartość atrybutu kolejki zablokowanej-get lub zablokowanej-put.

BŁĄD MQRC_INT_ATTR_COUNT_ERROR
(2021, X'7E5') Niepoprawna liczba atrybutów liczby całkowitej.

BŁĄD TABELI MQRC_INT_ATTRS_ARRAY_ERROR
(2023, X'7E7') Niepoprawna tablica atrybutów całkowitych.

MQRC_NOT_OPEN_FOR_SET (nieustawione)
(2040, X'7F8') Kolejka nie jest otwarta do ustawiania.

MQRC_OBJECT_CHANGED
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

MQRC_OBJECT_USZKODZONA
(2101, X'835 ') Obiekt uszkodzony.

BŁĄD STRONY MQRC_PAGESET_ERROR
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

MQRC_Q_DELETED (usunięto MQRC_Q_)
(2052, X'804 ') Kolejka została usunięta.

BŁĄD MQRC_Q_MGR_NAME_ERROR
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

MQRC_Q_MGR_NOT_AVAILABLE
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162, X'872 ') Menedżer kolejek jest zamykany.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_SELECTOR_COUNT_ERROR
(2065, X'811 ') Niepoprawna liczba selektorów.

BŁĄD WYWOŁANIA MQRC_SELECTOR_ERROR
(2067, X'813 ') Niepoprawny selektor atrybutu.

MQRC_SELECTOR_LIMIT_EXCEEDED
(2066, X'812 ') Zbyt duża liczba selektorów.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

MQRC_SUPPRESSED_BY_EXIT
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

BŁĄD MQRC_TRIGGER_CONTROL_ERROR
(2075, X'81B') Niepoprawna wartość atrybutu trigger-control.

MQRC_TRIGGER_DEPTH_ERROR,
(2076, X'81C') Niepoprawna wartość atrybutu trigger-depth.

MQRC_TRIGGER_MSG_PRIORITY_ERR
(2077, X'81D') Niepoprawna wartość atrybutu trigger-message-priority.

MQRC_TRIGGER_TYPE_ERROR (błąd typu wyzwalacza)
(2078, X'81E') Niepoprawna wartość atrybutu typu wyzwalacza.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Za pomocą tego wywołania aplikacja może określić tablicę atrybutów całkowitych, kolekcję łańcuchów atrybutów znakowych lub jedno i drugie. Jeśli nie wystąpią żadne błędy, wszystkie podane atrybuty są ustawiane równocześnie. Jeśli wystąpi błąd (na przykład, jeśli selektor jest niepoprawny lub podjęto próbę ustawienia atrybutu na niepoprawną wartość), wywołanie nie powiedzie się i nie zostaną ustawione żadne atrybuty.
2. Wartości atrybutów można określić za pomocą wywołania MQINQ. Szczegółowe informacje na ten temat zawiera sekcja “MQINQ-zapytanie o obiekt-atrybuty” na stronie 727 .

   Uwaga: Nie wszystkie atrybuty z wartościami, do których można uzyskać dostęp za pomocą wywołania MQINQ, mogą mieć zmienione wartości za pomocą wywołania MQSET. Na przykład za pomocą tego wywołania nie można ustawić żadnych atrybutów obiektu procesu ani menedżera kolejek.
3. Zmiany atrybutów są zachowywane po restarcie menedżera kolejek (inne niż zmiany tymczasowych kolejek dynamicznych, które nie są zachowywane po restarcie menedżera kolejek).
4. Nie można zmieniać atrybutów kolejki modelowej za pomocą wywołania MQSET. Jeśli jednak kolejka modelowa jest otwierana za pomocą wywołania MQOPEN z opcją MQOO_SET, można użyć wywołania MQSET do ustawienia atrybutów dynamicznej kolejki lokalnej utworzonej przez wywołanie MQOPEN.
5. Jeśli ustawiany obiekt jest kolejką klastra, musi istnieć lokalna instancja kolejki klastra, aby otwarcie powiodło się.

Więcej informacji na temat atrybutów obiektu zawiera sekcja:

- “Atrybuty kolejek” na stronie 863
- “Atrybuty listy nazw” na stronie 898
- “Atrybuty definicji procesów” na stronie 900
- “Atrybuty menedżera kolejek” na stronie 824

Wywołanie C

```
MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQSET' USING HCONN, HOBJ, SELECTORCOUNT, SELECTORS-TABLE,
                   INTATTRCOUNT, INTATTRS-TABLE, CHARATTRLENGTH,
                   CHARATTRS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQSET (Hconn, Hobj, SelectorCount, Selectors, IntAttrCount,
           IntAttrs, CharAttrLength, CharAttrs, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

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

Zadeklaruj parametry w następujący sposób:

```
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
```

Wywołanie Visual Basic

```
MQSET Hconn, Hobj, SelectorCount, Selectors, IntAttrCount, IntAttrs,
      CharAttrLength, CharAttrs, CompCode, Reason
```

Zadeklaruj parametry w następujący sposób:

```
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-ustawianie właściwości komunikatu

Wywołanie MQSETMP służy do ustawiania lub modyfikowania właściwości uchwytu komunikatu.

Składnia

MQSETMP (*Hconn*, *Hmsg*, *SetPropOpc*, *Nazwa*, *PropDesc*, *Typ*, *ValueLength*, *Wartość*, *Kod zakończenia*, *Przyczyna*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** . Jeśli uchwyt komunikatu został utworzony przy użyciu wywołania MQHC_UNASSOCIATED_HCONN, należy nawiązać poprawne połączenie w wątku, ustawiając właściwość uchwytu komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_CONNECTION_BROKEN.

Komunikat Hmsg
Typ: MQHMSG-input

Jest to uchwyt komunikatu, który ma zostać zmodyfikowany. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

SetPropOpcje
Typ: MQSMPO-input

Sterowanie sposobem ustawiania właściwości komunikatu.

Ta struktura umożliwia aplikacjom określanie opcji sterujących ustawianiem właściwości komunikatu. Struktura jest parametrem wejściowym wywołania MQSETMP. Więcej informacji na ten temat zawiera sekcja MQSMPO .

Nazwa
Typ: MQCHARV-input

Jest to nazwa właściwości, która ma zostać ustawiona.

Więcej informacji na temat używania nazw właściwości zawiera sekcja Nazwy właściwości i sekcja Ograniczenia dotyczące nazw właściwości .

PropDesc
Typ: MQPD-wejście/wyjście

Ta struktura jest używana do definiowania atrybutów właściwości, w tym:

- co się stanie, jeśli właściwość nie jest obsługiwana
- kontekst komunikatu, do którego należy właściwość
- Komunikaty, do których właściwość jest kopiowana podczas przepływu

Więcej informacji na temat tej struktury zawiera sekcja MQPD .

Typ
Typ: MQLONG-input

Typ danych ustawianej właściwości. Może to być jedna z następujących wartości:

MQTYPE_BOOLEAN (wartość boolowska)
Wartość boolowska. *ValueLength* musi mieć wartość 4.

MQTYPE_BYTE_STRING ŁAŃCUCH
Łańcuch bajtowy. *ValueLength* musi mieć wartość zero lub większą.

MQTYPE_INT8
8-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 1.

MQTYPE_INT16
16-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 2.

MQTYPE_INT32
32-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 4.

MQTYPE_INT64
64-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 8.

MQTYPE_FLOAT32
32-bitowa liczba zmiennopozycyjna. *ValueLength* musi mieć wartość 4.

Uwaga: ten typ nie jest obsługiwany w przypadku aplikacji korzystających z języka IBM COBOL for z/OS.

MQTYPE_FLOAT64
64-bitowa liczba zmiennopozycyjna. *ValueLength* musi mieć wartość 8.

Uwaga: ten typ nie jest obsługiwany w przypadku aplikacji korzystających z języka IBM COBOL for z/OS.

MQTYPE_STRING (łańcuch MQTYPE)
Łańcuch znaków. *ValueLength* musi mieć wartość zero lub większą albo wartość specjalną MQVL_NULL_TERMINATED.

MQTYPE_NULL
Właściwość istnieje, ale ma wartość null. *ValueLength* musi mieć wartość zero.

ValueLength
Typ: MQLONG-input

Długość (w bajtach) wartości właściwości w parametrze *wartość* . Zero jest poprawne tylko dla wartości null lub dla łańcuchów lub łańcuchów bajtowych. Wartość zero wskazuje, że właściwość istnieje, ale wartość nie zawiera żadnych znaków ani bajtów.

Wartość musi być większa lub równa zero lub następująca wartość specjalna, jeśli parametr *Type* ma ustawioną wartość MQTYPE_STRING:

MQVL_NULL_TERMINATED,
Wartość jest ograniczona przez pierwszą wartość null napotkaną w łańcuchu. Wartość NULL nie jest uwzględniana jako część łańcucha. Ta wartość jest niepoprawna, jeśli nie ustawiono również opcji MQTYPE_STRING.

Uwaga: Znak o kodzie zero używany do zakończenia łańcucha, jeśli parametr MQVL_NULL_TERMINATED jest ustawiony na wartość NULL z zestawu znaków wartości.

Wartość
Typ: MQBYTExValueLength-input

Wartość właściwości, która ma zostać ustawiona. Bufor musi być wyrównany do granicy odpowiedniej dla rodzaju danych w wartości.

W języku programowania C parametr jest zadeklarowany jako wskaźnik do unieważnienia; jako parametr można podać adres dowolnego typu danych.

Jeśli *ValueLength* ma wartość zero, *Wartość* nie jest przywoływana. W takim przypadku adres parametru przekazywany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość NULL.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

BŁĄD MQRC_RFH_FORMAT_ERROR
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'089C') Adapter nie jest dostępny.

BŁĄD MQRC_ADAPTER_SERV_LOAD_ERROR
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

MQRC_ASID_MISMATCH
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

BŁĄD MQRC_BUFFER_ERROR
(2004, X'07D4') Niepoprawny parametr wartości.

MQRC_BUFFER_LENGTH_ERROR
(2005, X'07D5') Niepoprawny parametr długości wartości.

MQRC_CALL_W_TOKU
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

BŁĄD MQRC_HMSG_ERROR
(2460, X'099C') Wskaźnik uchwytu komunikatu jest niepoprawny.

MQRC_MSG_HANDLE_IN_USE,
(2499, X'09C3') Uchwyt komunikatu jest już używany.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

BŁĄD MQRC_PD_ERROR
(2482, X'09B2') Niepoprawna struktura deskryptora właściwości.

BŁĄD MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Niepoprawna nazwa właściwości.

BŁĄD TYPU WŁAŚCIWOŚCI MQRC_PROPERTY_TYPE_ERROR
(2473, X'09A9') Niepoprawny typ danych właściwości.

MQRC_PROP_NUMBER_FORMAT_BŁĄD
(2472, X'09A8') Wystąpił błąd formatu liczb w danych wartości.

BŁĄD MQRC_SMPO_ERROR
(2463, X'099F') Niepoprawna struktura opcji ustawiania właściwości komunikatu.

BŁĄD MQRC_SOURCE_CCSID_ERROR
(2111, X'083F') Niepoprawny identyfikator kodowanego zestawu znaków nazwy właściwości.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Wywołanie C

```
MQSETMP (Hconn, Hmsg, &SetPropOpts, &Name, &PropDesc, Type,
ValueLength, &Value, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
     CALL 'MQSETMP' USING HCONN, HMSG, SETMSGOPTS, NAME, PROPDESC, TYPE,
                          VALUELENGTH, VALUE, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQSETMP (Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength,
              Value, CompCode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
         CALL MQSETMP,(HCONN,HMSG,SETMSGHOPTS,NAME,PROPDESC,TYPE,VALUELENGTH,
               VALUE,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-pobieranie informacji o statusie

Użyj wywołania MQSTAT, aby pobrać informacje o statusie. Typ zwracanych informacji o statusie jest określany przez wartość typu określoną w wywołaniu.

Składnia

MQSTAT (*Hconn*, *Typ*, *Stat*, *Compcode*, *Przyczyna*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Typ
Typ: MQLONG-input

Typ żądanych informacji o statusie. Poprawne wartości to:

MQSTAT_TYPE_ASYNC_ERROR
Zwraca informacje o poprzednich asynchronicznych operacjach umieszczania.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Zwraca informacje o ponownym połączeniu. Jeśli połączenie jest ponownie nawiązywane lub ponowne nawiązywanie połączenia nie powiodło się, informacje opisują niepowodzenie, które spowodowało rozpoczęcie ponownego nawiązywania połączenia.

Ta wartość jest poprawna tylko dla połączeń klienckich. W przypadku innych typów połączeń wywołanie kończy się niepowodzeniem z kodem przyczyny **MQRC_ENVIRONMENT_ERROR**

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Zwraca informacje o poprzednim niepowodzeniu związanym z ponownym nawiązaniem połączenia. Jeśli ponowne nawiązanie połączenia nie powiodło się, informacje opisują niepowodzenie, które spowodowało niepowodzenie ponownego nawiązania połączenia.

Ta wartość jest poprawna tylko dla połączeń klienckich. W przypadku innych typów połączeń wywołanie kończy się niepowodzeniem z kodem przyczyny **MQRC_ENVIRONMENT_ERROR**.

Status
Typ: MQSTS-wejście/wyjście

Struktura informacji o statusie. Szczegółowe informacje można znaleźć w sekcji “MQSTS-struktura raportowania statusu” na stronie 610.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_API_EXIT_ERROR
(2374, X' 946 ') Wyjście funkcji API nie powiodło się

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

MQRC_CALL_W_TOKU
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

ZERWANE POŁĄCZENIE MQRC_CONNECTION_BROKEN
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

ZATRZYMANE_POŁĄCZENIA_MQRC
(2203, X'89B') Połączenie jest zamykane.

MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

MQRC_Q_MGR_ZATRZYMYWANIE
(2162,X'872'-zatrzymywanie menedżera kolejek

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD TYPU MQRC_STAT_TYPE_ERROR
(2430, X'97E' Błąd typu MQSTAT

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_STS
(2426, X'97A') Błąd w strukturze MQSTS

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

1. Wywołanie funkcji MQSTAT , która określa typ operacji MQSTAT_TYPE_ASYNC_ERROR , zwraca informacje o poprzednich operacjach asynchronicznych MQPUT i MQPUT1 . Struktura MQSTS przekazana z powrotem z wywołania MQSTAT zawiera pierwsze zarejestrowane informacje o asynchronicznym ostrzeżeniu lub błędzie dla tego połączenia. Jeśli kolejne błędy lub ostrzeżenia następują po pierwszym, zwykle nie zmieniają one tych wartości. Jeśli jednak wystąpi błąd z kodem zakończenia MQCC_WARNING, zostanie zwrócona kolejna awaria z kodem zakończenia MQCC_FAILED .
2. Jeśli od momentu nawiązania połączenia lub od ostatniego wywołania metody MQSTAT nie wystąpiły żadne błędy, w strukturze MQSTS są zwracane wartości CompCode MQCC_OK i Reason of MQRC_NONE .
3. Liczba wywołań asynchronicznych, które zostały przetworzone pod uchwytem połączenia, jest zwracana przez trzy pola licznika: PutSuccessCount, PutWarningCount i PutFailureCount. Liczniki te są zwiększane przez menedżera kolejek za każdym razem, gdy operacja asynchroniczna jest przetwarzana pomyślnie, ma ostrzeżenie lub kończy się niepowodzeniem (należy pamiętać, że w celu rozliczania operacji umieszczania na liście dystrybucyjnej jest wykonywane raz dla kolejki docelowej, a nie raz dla listy dystrybucyjnej). Wartość licznika nie jest zwiększana poza maksymalną wartość dodatnią AMQ_LONG_MAX.
4. Pomyślne wywołanie funkcji MQSTAT powoduje zresetowanie wszystkich poprzednich informacji o błędach lub liczników.
5. Zachowanie parametru MQSTAT zależy od wartości parametru **MQSTAT Type** podanej przez użytkownika.
6. **MQSTAT_TYPE_ASYNC_ERROR**
    a. Wywołanie funkcji MQSTAT , która określa typ operacji MQSTAT_TYPE_ASYNC_ERROR , zwraca informacje o poprzednich operacjach asynchronicznych MQPUT i MQPUT1 . Struktura MQSTS przekazana z powrotem z wywołania MQSTAT zawiera pierwsze zarejestrowane informacje o asynchronicznym ostrzeżeniu lub błędzie dla tego połączenia. Jeśli kolejne błędy lub ostrzeżenia następują po pierwszym, zwykle nie zmieniają one tych wartości. Jeśli jednak wystąpi błąd z kodem zakończenia MQCC_WARNING, zostanie zwrócona kolejna awaria z kodem zakończenia MQCC_FAILED .
    b. Jeśli od momentu nawiązania połączenia lub od ostatniego wywołania metody MQSTAT nie wystąpiły żadne błędy, w strukturze MQSTS są zwracane wartości CompCode MQCC_OK i Reason of MQRC_NONE .

c. Liczba wywołań asynchronicznych, które zostały przetworzone pod uchwytem połączenia, jest zwracana przez trzy pola licznika: PutSuccessCount, PutWarningCount i PutFailureCount. Liczniki te są zwiększane przez menedżera kolejek za każdym razem, gdy operacja asynchroniczna jest przetwarzana pomyślnie, ma ostrzeżenie lub kończy się niepowodzeniem (należy pamiętać, że w celu rozliczania operacji umieszczania na liście dystrybucyjnej jest wykonywane raz dla kolejki docelowej, a nie raz dla listy dystrybucyjnej). Wartość licznika nie jest zwiększana poza maksymalną wartość dodatnią AMQ_LONG_MAX.

d. Pomyślne wywołanie funkcji MQSTAT powoduje zresetowanie wszystkich poprzednich informacji o błędach lub liczników.

MQSTAT_TYPE_RECONNECTION (typ MQSTAT_RECONNECTION)

Przypuśćmy, że podczas ponownego nawiązywania połączenia będzie używana funkcja MQSTAT z wartością Type ustawioną na MQSTAT_TYPE_RECONNECTION wewnątrz procedury obsługi zdarzeń. Należy wziąć pod uwagę poniższe przykłady.

Klient próbuje ponownie nawiązać połączenie lub nie powiodło się ponowne nawiązanie połączenia.
CompCode w strukturze MQSTS to MQCC_FAILED , a Reason może mieć wartość MQRC_CONNECTION_BROKEN lub MQRC_Q_MGR_QUIESCING. ObjectType ma wartość MQOT_Q_MGR, ObjectName jest nazwą menedżera kolejek, a ObjectQMgrName jest puste.

Klient pomyślnie zakończył ponowne połączenie lub nigdy nie został rozłączony.
CompCode w strukturze MQSTS to MQCC_OK , a Reason to MQRC_NONE

Kolejne wywołania funkcji MQSTAT zwracają te same wyniki.

BŁĄD PONOWNEGO połączenia MQSTAT_TYPE_RECONNECTION_ERROR

Przypuśćmy, że użytkownik wywoła funkcję MQSTAT z wartością Type ustawioną na MQSTAT_TYPE_RECONNECTION_ERROR w odpowiedzi na odebranie MQRC_RECONNECT_FAILED na wywołanie MQI. Należy wziąć pod uwagę poniższe przykłady.

Wystąpił błąd autoryzacji podczas ponownego otwierania kolejki podczas ponownego nawiązywania połączenia z innym menedżerem kolejek.
CompCode w strukturze MQSTS ma wartość MQCC_FAILED , a Reason jest przyczyną niepowodzenia ponownego połączenia, na przykład MQRC_NOT_AUTHORIZED. ObjectType to typ obiektu, który spowodował problem, na przykład MQOT_QUEUE, ObjectName to nazwa kolejki i ObjectQMgrName nazwa menedżera kolejek będącego właścicielem kolejki.

Wystąpił błąd połączenia gniazda podczas ponownego połączenia.
CompCode w strukturze MQSTS ma wartość MQCC_FAILED , a Reason jest przyczyną niepowodzenia ponownego połączenia, na przykład MQRC_HOST_NOT_AVAILABLE. ObjectType ma wartość MQOT_Q_MGR, ObjectName jest nazwą menedżera kolejek, a ObjectQMgrName jest puste.

Kolejne wywołania funkcji MQSTAT zwracają te same wyniki.

Wywołanie C

```
MQSTAT (Hconn, StatType, &Stat, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN Hconn;        /* Connection Handle */
MQLONG StatType;    /* Status type */
MQSTS Stat;           /* Status information structure */
MQLONG CompCode;    /* Completion code */
MQLONG Reason;        /* Reason code qualifying CompCode */
```

Wywołanie COBOL

```
CALL 'MQSTAT' USING HCONN, STATTYPE, STAT, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**    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.
```

Wywołanie PL/I

```
call MQSTAT (Hconn, StatType, Stat, Compcode, Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie asemblera System/390

```
CALL MQSTAT,(HCONN,STATTYPE,STAT,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-rejestrowanie subskrypcji

Wywołanie MQSUB służy do rejestrowania subskrypcji aplikacji w konkretnym temacie.

Składnia

MQSUB (*Hconn*, *SubDesc*, *Hobj*, *Hsub*, *Compcode*, *Reason*)

Parametry

hconn

Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

SubDesc
Typ: MQSD-input/output

Jest to struktura identyfikująca używany obiekt, który jest rejestrowany przez aplikację. Więcej informacji zawiera sekcja "MQSD-deskryptor subskrypcji" na stronie 584.

Hobby
Typ: MQHOBJ-input/output

Ten uchwyt reprezentuje dostęp, który został ustanowiony w celu uzyskania komunikatów wysłanych do tej subskrypcji. Komunikaty te mogą być składowane w konkretnej kolejce lub menedżer kolejek może zarządzać pamięcią masową bez używania konkretnej kolejki.

Aby użyć konkretnej kolejki, należy powiązać ją z subskrypcją podczas tworzenia subskrypcji. Można to zrobić na dwa sposoby:

- Używając komendy MQSC DEFINE SUB i udostępniając tę komendę z nazwą obiektu kolejki.
- Przez podanie tego uchwytu podczas wywoływania MQSUB z opcją MQSO_CREATE

  Jeśli ten uchwyt jest podany jako parametr wejściowy w wywołaniu, musi być poprawnym uchwytem obiektu zwróconym z poprzedniego wywołania MQOPEN kolejki przy użyciu co najmniej jednej z następujących opcji:

  – MQOO_INPUT_ *
  – MQOO_BROWSE,
  – MQOO_OUTPUT (jeśli kolejka jest kolejką zdalną)

  W przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_HOBJ_ERROR. Nie może to być uchwyt obiektu do kolejki aliasowej, która jest tłumaczona na obiekt tematu. Jeśli tak, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_HOBJ_ERROR.

Jeśli menedżer kolejek ma zarządzać przechowywaniem komunikatów wysyłanych do tej subskrypcji, należy to ustawić podczas tworzenia subskrypcji przy użyciu opcji MQSO_MANAGED. Następnie menedżer kolejek zwraca ten uchwyt jako parametr wyjściowy w wywołaniu. Zwrócony uchwyt jest nazywany uchwytem zarządzanym. Jeśli określono parametr MQHO_NONE, ale nie określono parametru MQSO_MANAGED, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_HOBJ_ERROR.

Po zwróceniu zarządzanego uchwytu przez menedżer kolejek można go użyć w wywołaniu MQGET lub MQCB z opcjami przeglądania lub bez nich, w wywołaniu MQINQ lub w wywołaniu MQCLOSE. Nie można jej użyć w MQPUT, MQSUB, MQSET; próba wykonania tej operacji kończy się niepowodzeniem z błędem MQRC_NOT_OPEN_FOR_OUTPUT, MQRC_HOBJ_ERROR lub MQRC_NOT_OPEN_FOR_SET.

Jeśli ta subskrypcja jest wznawiana przy użyciu opcji MQSO_RESUME w strukturze MQSD, uchwyt może zostać zwrócony do aplikacji w tym parametrze przez ustawienie parametru MQSO_MANAGED na wartość MQHO_NONE. Można to zrobić niezależnie od tego, czy subskrypcja używa zarządzanego uchwytu, czy nie, i może być przydatne udostępnienie subskrypcji utworzonych za pomocą komendy DEFINE SUB z uchwytem do kolejki subskrypcji zdefiniowanej w tej komendzie. Jeśli subskrypcja utworzona administracyjnie jest wznawiana, kolejka jest otwierana za pomocą komend MQOO_INPUT_AS_Q_DEF i MQOO_BROWSE. Jeśli konieczne jest określenie innych opcji, aplikacja musi jawnie otworzyć kolejkę subskrypcji i udostępnić uchwyt obiektu w wywołaniu. Jeśli wystąpi problem z otwarciem kolejki, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_INVALID_DESTINATION. Jeśli podano parametr *Hobj* , musi on być odpowiednikiem parametru *Hobj* w oryginalnym wywołaniu MQSUB. Oznacza to, że jeśli udostępniany jest uchwyt obiektu zwrócony przez wywołanie MQOPEN, uchwyt musi znajdować się w tej samej kolejce, co

poprzednio używane. Jeśli nie jest to ta sama kolejka, wywołanie nie powiedzie się i zostanie wyświetlony komunikat MQRC_HOBJ_ERROR.

Jeśli ta subskrypcja jest zmieniana za pomocą opcji MQSO_ALTER w strukturze MQSD, można podać inną wartość *Hobj* . Wszystkie publikacje, które zostały dostarczone do kolejki i zostały wcześniej zidentyfikowane za pomocą tego parametru, pozostają w tej kolejce i aplikacja jest odpowiedzialna za pobranie tych komunikatów, jeśli parametr **Hobj** reprezentuje teraz inną kolejkę.

Tabela 555. Korzystanie z hobj z różnymi opcjami subskrypcji

Opcje	***Hobj***	**Opis**
MQSO_CREATE + MQSO_MANAGED	Zignorowano na wejściu	Tworzy subskrypcję z pamięcią masową komunikatów zarządzanych przez menedżer kolejek
MQSO_CREATE	Poprawny uchwyt obiektu	Tworzy subskrypcję udostępniającą konkretną kolejkę jako miejsce docelowe dla komunikatów.
MQSO_RESUME	MQHO_NONE	Wznawia wcześniej utworzoną subskrypcję, niezależnie od tego, czy była zarządzana, czy nie, i zwraca uchwyt obiektu do użycia przez aplikację przez menedżer kolejek.
MQSO_RESUME	Poprawny, zgodny, uchwyt obiektu	Wznawia utworzoną wcześniej subskrypcję, która używa określonej kolejki jako miejsca docelowego dla komunikatów i używa uchwytu obiektu z konkretnymi opcjami otwarcia.
MQSO_ALTER + MQSO_MANAGED	MQHO_NONE	Zmienia istniejącą subskrypcję, która wcześniej używała konkretnej kolejki, więc jest teraz subskrypcją zarządzaną. Nie można zmienić klasy docelowej (zarządzanej lub nie).
MQSO_ALTER (zmiana MQSO)	Poprawny uchwyt obiektu	Zmienia istniejącą subskrypcję, niezależnie od tego, czy była zarządzana, czy nie, tak aby używała konkretnej kolejki. Jeśli opcja MQSO_MANAGED nie jest używana, można zmienić podaną kolejkę, ale nie można zmienić klasy miejsca docelowego (zarządzanego lub nie).

Niezależnie od tego, czy została ona udostępniona, czy zwrócona, parametr *Hobj* musi być określony w kolejnych wywołaniach MQGET lub MQCB, które mają odbierać komunikaty publikacji wysyłane do tej subskrypcji.

Uchwyt *Hobj* nie jest już poprawny, gdy zostanie dla niego wykonane wywołanie MQCLOSE lub gdy jednostka przetwarzania definiująca zasięg uchwytu zostanie zakończona (dopóki aplikacja nie rozłączy się). Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Informacje na temat zasięgu uchwytu zawiera sekcja Hconn (MQHCONN)-wyjście . Operacja MQCLOSE dla uchwytu *Hobj* nie ma wpływu na uchwyt *Hsub* .

Hsub
Typ: MQHOBJ-wyjście

Ten uchwyt reprezentuje subskrypcję, która została wykonana. Może być używany do dwóch dodatkowych operacji:

- Można go użyć w kolejnym wywołaniu MQSUBRQ w celu wysłania żądania wysłania publikacji, gdy podczas tworzenia subskrypcji użyto opcji MQSO_PUBLIC ATIONS_ON_REQUEST.
- Można go użyć w kolejnym wywołaniu MQCLOSE w celu usunięcia subskrypcji, która została wykonana. Uchwyt *Hsub* przestaje być poprawny w przypadku wywołania MQCLOSE lub zakończenia jednostki przetwarzania definiującej zasięg uchwytu. Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Operacja MQCLOSE dla uchwytu *Hsub* nie ma wpływu na uchwyt *Hobj* .

Tego uchwytu nie można przekazać do wywołania MQGET lub MQCB. Należy użyć parametru **Hobj** . Tego uchwytu nie można używać w żadnym wywołaniu programu IBM MQ innym niż MQCLOSE lub MQSUBRQ. Przekazanie tego uchwytu do dowolnego innego wywołania IBM MQ powoduje błąd MQRC_HOBJ_ERROR.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Pomyślne zakończenie

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK, kod przyczyny jest następujący:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED, kod przyczyny ma jedną z następujących wartości:

BŁĄD MQRC_CLUSTER_RESOLUTION_ERROR
(2189, X'88D') Rozstrzyganie nazwy klastra nie powiodło się.

MQRC_DURABILITY_NOT_ALLOWED
2436 (X'0984 ') Wywołanie MQSUB z użyciem opcji MQSO_DURABLE nie powiodło się.

MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)
2298 (X'08FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

BŁĄD MQRC_HOBJ_ERROR
2019 (X'07E3') Niepoprawny uchwyt obiektu Hobj.

MQRC_IDENTITY_MISMATCH
2434 (X'0982 ') Nazwa subskrypcji jest zgodna z istniejącą subskrypcją.

MQRC_NOT_AUTHORIZED (nieautoryzowany)
2035 (X'07F3') Użytkownik nie ma uprawnień do wykonania operacji.

MQRC_NO_SUBSCRIPTION
2428 (X'097C') Podana nazwa subskrypcji nie istnieje.

BŁĄD WYWOŁANIA MQRC_OBJECT_STRING_ERROR
Pole łańcucha obiektu 2441 (X'0989 ') jest niepoprawne.

BŁĄD MQRC_OPTIONS_ERROR
Parametr lub pole opcji 2046 (X'07FE') zawiera niepoprawne opcje lub kombinację niepoprawnych opcji.

MQRC_Q_MGR_QUIESCING,
2161 (X'0871 ') wygaszanie menedżera kolejek.

MQRC_RECONNECT_Q_MGR_REQD
2555 (X'09FB' X) Opcja MQCNO_RECONNECT_Q_MGR jest wymagana.

BŁĄD MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') Nie można pobrać zachowanych publikacji, które istnieją dla zasubskrybowanego łańcucha tematu.

MQRC_RETAINED_NOT_DOSTARCZONE
2526 (X'09DE') Zachowane publikacje, które istnieją dla zasubskrybowanego łańcucha tematu, nie mogą zostać dostarczone do kolejki docelowej subskrypcji i nie mogą zostać dostarczone do kolejki niedostarczonych komunikatów.

BŁĄD MQRC_SD_ERROR
2424 (X'0978 ') Niepoprawny deskryptor subskrypcji (MQSD).

MQRC_SELECTION_NOT_AVAILABLE
2551 (X'09F7') Łańcuch wyboru nie jest zgodny ze składnią selektora IBM MQ i nie był dostępny żaden rozszerzony dostawca wyboru komunikatów.

Błąd łańcucha MQRC_SELECTION_STRING_ERROR
2519 (X'09D7') Łańcuch wyboru musi być określony zgodnie z opisem w dokumentacji struktury MQCHARV.

BŁĄD SYNTAX_MQRC_SELECTOR_SYNTAX_ERROR
2459 (X'099B') Wywołanie MQOPEN, MQPUT1lub MQSUB zostało wykonane, ale określono instrukcję wyboru, która zawierała błąd składniowy.

MQRC_SUB_USER_DATA_BŁĄD
Pole danych 2431 (X'097F') SubUserjest niepoprawne.

BŁĄD MQRC_SUB_NAME_ERROR
Pole 2440 (X'0988 ') SubName jest niepoprawne.

MQRC_SUB_ALREADY_EXISTS
2432 (X'0980 ') Subskrypcja już istnieje.

MQRC_SUB_USER_DATA_BŁĄD
Pole danych 2431 (X'097F') SubUserjest niepoprawne.

BŁĄD MQRC_TOPIC_STRING_ERROR
2425 (X'0979 ') Łańcuch tematu jest niepoprawny.

MQRC_UNKNOWN_OBJECT_NAME
2085 (X'0825 ') Nie można znaleźć obiektu zidentyfikowanego w polu ObjectName MQSD.

MQRC_SUB_JOIN_NOT_ALTERABLE (NIE MOŻNA)
29440 (X'7300 ') Tryb współużytkowania subskrypcji jest niezgodny z istniejącą subskrypcją. Ten błąd może zostać zwrócony podczas próby wznowienia współużytkowanej subskrypcji JMS 2.0 w aplikacji innej niż JMS.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

- Subskrypcja tematu jest wykonywana na podstawie nazwy skróconej predefiniowanego obiektu tematu, pełnej nazwy łańcucha tematu lub jest tworzona przez konkatenację dwóch części. Patrz opis *ObjectName* i *ObjectString* w sekcji “MQSD-deskryptor subskrypcji” na stronie 584.
- Menedżer kolejek przeprowadza sprawdzanie zabezpieczeń w momencie wywołania MQSUB w celu sprawdzenia, czy identyfikator użytkownika, pod którym działa aplikacja, ma odpowiedni poziom uprawnień przed zezwoleniem na dostęp. Odpowiedni obiekt tematu znajduje się w hierarchii tematów i jest wykonywane sprawdzanie uprawnień do tego obiektu tematu, aby zapewnić ustawienie uprawnień

do subskrybowania. Jeśli opcja MQSO_MANAGED nie jest używana, w kolejce docelowej wykonywane jest sprawdzenie uprawnień, aby upewnić się, że uprawnienie do danych wyjściowych jest ustawione. Jeśli używana jest opcja MQSO_MANAGED, nie jest wykonywane sprawdzanie uprawnień w kolejce zarządzanej w celu uzyskania dostępu do danych wyjściowych lub zapytań.

- Jeśli jako dane wejściowe nie podano Hobj, wywołanie MQSUB przydziela dwa uchwyty, uchwyt obiektu (Hobj) i uchwyt subskrypcji (Hsub).
- Obiekt Hobj zwrócony w wywołaniu MQSUB, gdy używana jest opcja MQSO_MANAGED, może zostać sprawdzony w celu znalezienia atrybutów, takich jak próg wycofania i nadmierna nazwa kolejki wycofania. Można również zapytać o nazwę kolejki zarządzanej, ale nie można próbować bezpośrednio otworzyć tej kolejki.
- Subskrypcje mogą być grupowane, dzięki czemu tylko jedna publikacja może zostać dostarczona do grupy subskrypcji, nawet jeśli więcej niż jedna grupa jest zgodna z daną publikacją. Subskrypcje są grupowane przy użyciu opcji MQSO_GROUP_SUB i w celu grupowania subskrypcji muszą być
  - przy użyciu tej samej kolejki nazwanej (która nie korzysta z opcji MQSO_MANAGED) w tym samym menedżerze kolejek-reprezentowanym przez parametr Hobj w wywołaniu MQSUB
  - współużytkuj ten sam identyfikator SubCorrel
  - być tego samego SubLevel

  Te atrybuty definiują zestaw subskrypcji uważanych za należące do grupy, a także są atrybutami, których nie można zmienić, jeśli subskrypcja jest zgrupowana. Zmiana SubLevel powoduje zmianę wartości MQRC_SUBLEVEL_NOT_ALTERABLE, a zmiana innych wartości (którą można zmienić, jeśli subskrypcja nie jest pogrupowana) powoduje, że MQRC_GROUPING_NOT_ALTERABLE.
- Pomyślne zakończenie wywołania MQSUB nie oznacza, że działanie zostało zakończone. Aby sprawdzić, czy wywołanie zostało zakończone, należy zapoznać się z krokiem DEFINE SUB w sekcji Sprawdzanie, czy komendy asynchroniczne dla sieci rozproszonych zostały zakończone.
- Pola w danych MQSD są wypełniane po powrocie z wywołania MQSUB, które korzysta z opcji MQSO_RESUME. Zwrócony kod MQSD może zostać przekazany bezpośrednio do wywołania MQSUB, które używa opcji MQSO_ALTER z dowolnymi zmianami, które należy wprowadzić w subskrypcji zastosowanej do pliku MQSD. Niektóre pola mają specjalne uwagi, jak zaznaczono w tabeli.

Tabela 556. Specjalne uwagi dotyczące pól w produkcie MQSD

Nazwa pola w MQSD	**Uwagi szczególne**
Opcje dostępu lub tworzenia	Niektóre opcje mogą zostać zresetowane po powrocie z wywołania MQSUB. Jeśli usługa MQSD będzie ponownie wykorzystywana w wywołaniu MQSUB, wymagana opcja musi być jawnie ustawiona.
Opcje trwałości, opcje docelowe, opcje rejestracji i znaki wieloznaczne	Te opcje są odpowiednio ustawione
Opcje publikacji	Te opcje są ustawiane zgodnie z potrzebami, z wyjątkiem opcji MQSO_NEW_PUBLIC, które mają zastosowanie tylko do opcji MQSO_CREATE.
Inne opcje	Te opcje nie ulegają zmianie po powrocie z wywołania MQSUB. Sterują one sposobem wywoływania interfejsu API i nie są przechowywane z subskrypcją. Muszą one zostać ustawione zgodnie z wymaganiami w każdym kolejnym wywołaniu MQSUB z ponownym wykorzystaniem usługi MQSD.
ObjectName	To pole wejściowe pozostaje niezmienione w przypadku powrotu z wywołania MQSUB.

Tabela 556. Specjalne uwagi dotyczące pól w produkcie MQSD (kontynuacja)

Nazwa pola w MQSD	**Uwagi szczególne**
ObjectString	To pole wejściowe pozostaje niezmienione w przypadku powrotu z wywołania MQSUB. Jeśli zostanie podany bufor, w polu *ResObjectString* zwracana jest pełna nazwa tematu.
AlternateUserId i AlternateSecurityId	Te pola wejściowe są niezmienione w przypadku powrotu z wywołania MQSUB. Sterują one sposobem wywoływania interfejsu API i nie są przechowywane z subskrypcją. Muszą one zostać ustawione zgodnie z wymaganiami w każdym kolejnym wywołaniu MQSUB używającego ponownie usługi MQSD.
SubExpiry	Po powrocie z wywołania MQSUB przy użyciu opcji MQSO_RESUME w tym polu jest ustawiana pierwotna utrata ważności subskrypcji, a nie pozostały czas utraty ważności. Jeśli usługa MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB przy użyciu opcji MQSO_ALTER, utrata ważności subskrypcji zostanie zresetowana w celu ponownego rozpoczęcia zliczania.
SubName	To pole jest polem wejściowym wywołania MQSUB i nie jest zmieniane na wyjściu.
SubUserData i SelectionString	Te pola o zmiennej długości są zwracane w danych wyjściowych wywołania MQSUB przy użyciu opcji MQSO_RESUME (jeśli podano bufor), a także dodatniej długości buforu w produkcie *VSBufSize*. Jeśli nie podano żadnego buforu, zwracana jest tylko długość w polu *VSLength* tabeli MQCHARV. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w podanym buforze zwracane są tylko *VSBufSize* bajtów. Jeśli usługa MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB z użyciem opcji MQSO_ALTER, a bufor nie zostanie udostępniony, ale zostanie podana wartość niezerowa *VSLength* , to jeśli ta długość będzie zgodna z istniejącą długością pola, pole nie zostanie w nim wprowadzone żadne zmiany.
SubCorrelId i znacznik PubAccounting	Jeśli identyfikator MQSO_SET_CORREL_ID nie jest używany, identyfikator *SubCorrelId* jest generowany przez menedżer kolejek. Jeśli parametr MQSO_SET_IDENTITY_CONTEXT nie zostanie użyty, menedżer kolejek wygeneruje plik *PubAccountingToken* . Te pola są zwracane w danych MQSD z wywołania MQSUB przy użyciu opcji MQSO_RESUME. Jeśli są one generowane przez menedżer kolejek, wygenerowana wartość jest zwracana w wywołaniu MQSUB przy użyciu opcji MQSO_CREATE lub MQSO_ALTER.
PubPriority, SubLevel & PubApplIdentityData	Te pola są zwracane w danych MQSD.
Łańcuch ResObject	To pole tylko wyjściowe jest zwracane w pliku MQSD, jeśli udostępniono bufor.

Wywołanie C

```
MQSUB (Hconn, &SubDesc, &Hobj, &Hsub, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQSUB' USING HCONN, SUBDESC, HOBJ, HSUB, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Wywołanie PL/I

```
call MQSUB (Hconn, SubDesc, Hobj, Hsub, CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQSUB,(HCONN,SUBDESC,HOBJ,HSUB,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-żądanie subskrypcji

Wywołanie MQSUBRQ służy do żądania zachowanej publikacji, gdy subskrybent został zarejestrowany w MQSO_PUBLIC S_ON_REQUEST.

Składnia

MQSUBRQ (*Hconn*, *Hsub*, *Action*, *SubRqOpts*, *Compcode*, *Reason*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *Hconn*:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Hsub
Typ: MQHOBJ-input

Ten uchwyt reprezentuje subskrypcję, dla której ma zostać zażądana aktualizacja. Wartość *Hsub* została zwrócona z poprzedniego wywołania MQSUB.

Działanie
Typ: MQLONG-input

Ten parametr steruje konkretnym działaniem, które jest żądane dla subskrypcji. Należy podać następującą wartość:

PUBLIKACJA_DZIAŁANIA_MQSRR

To działanie powoduje wysłanie żądania aktualizacji publikacji dla określonego tematu. Można go używać tylko wtedy, gdy subskrybent określił opcję MQSO_PUBLIC ATIONS_ON_REQUEST w wywołaniu MQSUB podczas tworzenia subskrypcji. Jeśli menedżer kolejek ma zachowaną publikację dla tematu, jest ona wysyłana do subskrybenta. Jeśli nie, wywołanie nie powiedzie się. Jeśli do aplikacji zostanie wysłana zachowana publikacja, jest to wskazywane przez właściwość komunikatu MQIsRetained tej publikacji.

Ponieważ temat w istniejącej subskrypcji reprezentowany przez parametr Hsub może zawierać znaki wieloznaczne, subskrybent może otrzymać wiele zachowanych publikacji.

SubRqOpcje
Typ: MQSRO-input/output

Te opcje sterują działaniem programu MQSUBRQ. Szczegółowe informacje na ten temat zawiera sekcja “MQSRO-opcje żądania subskrypcji” na stronie 607 .

Jeśli nie są wymagane żadne opcje, programy napisane w języku C lub S/390 mogą określić pusty adres parametru zamiast adresu struktury MQSRO.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia; jest to jeden z następujących kodów:

MQCC_OK
Pomyślne zakończenie

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)
2298 (X'08FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

MQRC_NO_RETAINED_MSG
2437 (X'0985 ') Dla tego tematu nie są obecnie przechowywane żadne zachowane publikacje.

BŁĄD MQRC_OPTIONS_ERROR
Parametr lub pole opcji 2046 (X'07FE') zawiera niepoprawne opcje lub kombinację niepoprawnych opcji.

MQRC_Q_MGR_QUIESCING,
2161 (X'0871 ') wygaszanie menedżera kolejek.

BŁĄD MQRC_SRO_ERROR
2438 (X'0986 ') W wywołaniu MQSUBRQ opcje żądania subskrypcji MQSRO są niepoprawne.

BŁĄD MQRC_RETAINED_MSG_Q_ERROR
2525 (X'09DD') Nie można pobrać zachowanych publikacji, które istnieją dla zasubskrybowanego łańcucha tematu.

MQRC_RETAINED_NOT_DOSTARCZONE
2526 (X'09DE') Zachowane publikacje, które istnieją dla zasubskrybowanego łańcucha tematu, nie mogą zostać dostarczone do kolejki docelowej subskrypcji i nie mogą zostać dostarczone do kolejki niedostarczonych komunikatów.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Użycie notatek

Poniższe uwagi dotyczące użycia dotyczą użycia kodu działania MQSR_ACTION_PUBLICATION:

1. Jeśli ta komenda zakończy się pomyślnie, zachowane publikacje zgodne z określoną subskrypcją zostaną wysłane do subskrypcji i mogą zostać odebrane za pomocą komendy MQGET lub MQCB przy użyciu Hobj zwróconego w oryginalnej komendzie MQSUB, która utworzyła subskrypcję.
2. Jeśli temat zasubskrybowany przez pierwotną komendę MQSUB, która utworzyła subskrypcję, zawiera znak wieloznaczny, można wysłać więcej niż jedną zachowaną publikację. Liczba publikacji wysłanych w wyniku tego wywołania jest rejestrowana w polu NumPubs w strukturze opcji SubRq.
3. Jeśli komenda zostanie zakończona z kodem przyczyny MQRC_NO_RETAINED_MSG, oznacza to, że nie ma obecnie zachowanych publikacji dla podanego tematu. #
4. Jeśli komenda zakończy działanie z kodem przyczyny MQRC_RETAINED_MSG_Q_ERROR lub MQRC_RETAINED_NOT_DOSTARCZONE, istnieją obecnie zachowane publikacje dla określonego tematu, ale wystąpił błąd, który oznacza, że nie można ich dostarczyć.
5. Aplikacja musi mieć bieżącą subskrypcję tematu, zanim będzie mogła wykonać to wywołanie. Jeśli subskrypcja została utworzona w poprzedniej instancji aplikacji i nie jest dostępny poprawny uchwyt subskrypcji, aplikacja musi najpierw wywołać MQSUB z opcją MQSO_RESUME, aby uzyskać uchwyt do użycia w tym wywołaniu.
6. Publikacje są wysyłane do miejsca docelowego, które jest zarejestrowane do użytku z bieżącą subskrypcją tej aplikacji. Jeśli publikacje muszą zostać wysłane w innym miejscu, należy najpierw zmienić subskrypcję za pomocą wywołania MQSUB z opcją MQSO_ALTER.

Wywołanie C

```
MQSUB (Hconn, Hsub, Action, &SubRqOpts, &CompCode, &Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
CALL 'MQSUBRQ' USING HCONN, HSUB, ACTION, SUBRQOPTS, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
** 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.
```

Wywołanie PL/I

```
call MQSUBRQ (Hconn, Hsub, Action, SubRqOpts, CompCode, Reason)
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Wywołanie programu High Level Assembler

```
CALL MQSUBRQ,(HCONN, HSUB, ACTION, SUBRQOPTS,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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
```

Atrybuty obiektów

W tej kolekcji tematów wymieniono tylko te obiekty IBM MQ , które mogą być przedmiotem wywołania funkcji MQINQ, oraz podano szczegółowe informacje o atrybutach, do których można tworzyć zapytania, i selektorach, które mają być używane.

Atrybuty menedżera kolejek

Niektóre atrybuty menedżera kolejek są stałe dla konkretnych implementacji, inne można zmienić za pomocą komendy MQSC ALTER QMGR.

Atrybuty można również wyświetlić za pomocą komendy DISPLAY QMGR. Większość atrybutów menedżera kolejek można uzyskać, otwierając specjalny obiekt MQOT_Q_MGR i używając wywołania MQINQ ze zwróconym uchwytem.

W poniższej tabeli znajduje się podsumowanie atrybutów specyficznych dla menedżera kolejek. Atrybuty są opisane w porządku alfabetycznym.

Uwaga: Nazwy atrybutów przedstawione w tej sekcji są opisowymi nazwami używanymi w wywołaniu MQINQ; nazwy są takie same, jak w przypadku komend PCF. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja Komendy MQSC .

Tabela 557. Atrybuty menedżera kolejek

Atrybut	Opis
AccountingConnNadpisz	Nadpisz ustawienia rozliczania.
AccountingInterval	Częstotliwość zapisywania pośrednich rekordów rozliczeniowych.
NadpisanieActivityConn	Nadpisz ustawienia działania.
ActivityTrace	Steruje gromadzeniem danych śledzenia działań aplikacji MQI produktu IBM MQ .
AdoptNewMCACheck	Elementy sprawdzone w celu określenia, czy ma zostać przyjęte nowe MCA.
AdoptNewMCAType	Określa, czy automatycznie restartować osieroconą instancję agenta MCA określonego typu kanału.
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
AuthorityEvent	Określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane)
BridgeEvent	Atrybut sterujący dla zdarzeń mostu.
ChannelAutoOdw	Określa, czy dozwolona jest automatyczna definicja kanału
ChannelAutoDefEvent	Określa, czy generowane są zdarzenia automatycznej definicji kanału
ChannelAutoDefExit	Nazwa procedury zewnętrznej dla automatycznej definicji kanału
ChannelEvent	Atrybut sterujący dla zdarzeń kanału.
Element sterującyChannelInitiator	Atrybut sterujący inicjatora kanału
ChannelMonitoring	Dane monitorowania bezpośredniego dla kanałów
ChannelStatistics	Steruje gromadzeniem danych statystycznych dla kanałów.
ChinitAdapters	Liczba podzadań adaptera do przetwarzania wywołań IBM MQ .
ChinitDispatchers	Liczba programów rozsyłających, które mają być używane dla inicjatora kanału.
	Zarezerwowane do użytku przez IBM .
ChinitTraceAutoStart	Określa, czy śledzenie inicjatora kanału ma być uruchamiane automatycznie.
ChinitTraceTableSize	Wielkość obszaru danych śledzenia inicjatora kanału.
ClusterSenderMonitoringDefault	Domyślne dane monitorowania bezpośredniego dla kanałów wysyłających klastry
ClusterSenderClusterSender	Steruje gromadzeniem informacji o monitorowaniu statystyk dla kanałów nadajnika klastra.
ClusterWorkloadDane	Dane użytkownika dla wyjścia obciążenia klastra
ClusterWorkloadWyjdź	Nazwa programu zewnętrznego dla zarządzania obciążeniem klastra
ClusterWorkloadDługość	Maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra

Tabela 557. Atrybuty menedżera kolejek (kontynuacja)	
Atrybut	**Opis**
Kanały CLWLMRUChannel	Liczba ostatnio używanych kanałów na potrzeby równoważenia obciążenia klastra
CLWLUseQ	Obciążenie klastra używa kolejki zdalnej.
CodedCharSetId	Identyfikator kodowanego zestawu znaków
CommandEvent	Atrybut sterujący dla zdarzeń komendy.
CommandInputQName, atrybut	Nazwa kolejki wejściowej komendy
CommandLevel	Poziom komendy
CommandServerControl, atrybut	Atrybut sterujący dla serwera komend.
atrybut zdarzenia konfiguracji	Atrybut sterujący dla zdarzeń konfiguracji.
DeadLetterQName	Nazwa kolejki niedostarczonych komunikatów
DefClusterXmitQueueTyp	Domyślny typ kolejki transmisji klastra
DefXmitQName	Domyślna nazwa kolejki transmisji
DistLists	Obsługa listy dystrybucyjnej
DNSGROUP	Nazwa grupy nasłuchiwania TCP w przypadku korzystania z obsługi dynamicznych usług nazw domen (DNS) programu Workload Manager.
DNSWLM	Określa, czy program nasłuchujący TCP rejestruje się w programie Workload Manager dla usług Dynamic Domain Name Services.
ExpiryInterval	Odstęp czasu między operacjami skanowania komunikatów, które utraciły ważność
IGQPutAuthority	Uprawnienie do umieszczania w kolejce wewnątrz grupy
IGQUserId	Identyfikator użytkownika kolejkowania wewnątrz grupy
InhibitEvent	Określa, czy są generowane zdarzenia blokady (zablokuj pobieranie i zablokuj umieszczanie)
V 9.3.0 V 9.3.0 InitialKey 1	Klucz początkowy dla systemu zabezpieczenia hasłem.
IPAddressVersion	Wersja adresu Internet Protocol
IntraGroupqueuing	Obsługa kolejkowania wewnątrz grupy
ListenerTimer	Odstęp czasu między próbami zrestartowania programu nasłuchującego po awarii APPC lub TCP/IP.
LocalEvent	Określa, czy generowane są lokalne zdarzenia błędów
LoggerEvent	Określa, czy generowane są zdarzenia programu rejestrującego
LUGroupName	Ogólna nazwa jednostki logicznej dla programu nasłuchującego LU 6.2 , który obsługuje transmisje przychodzące dla grupy współużytkowania kolejek.
LUNAME	Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 .
LU62ARMSuffix	Przyrostek SYS1.PARMLIB podzbiór APPCPMxx, który wyznacza LUADD dla tego inicjatora kanału.
LU62Channels	Maksymalna liczba bieżących kanałów lub podłączonych klientów używających jednostki logicznej 6.2.
MaxActiveChannels	Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie.
MaxChannels	Maksymalna liczba bieżących kanałów.
MaxHandles	Maksymalna liczba uchwytów
MaxMsgdługość komunikatu	Maksymalna długość komunikatu w bajtach
atrybutMaxPriority	Maksymalny priorytet
MaxPropertiesDługość	Maksymalna długość danych właściwości w bajtach
MaxUncommittedKomunikaty	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy
MQIAccounting	Steruje gromadzeniem informacji rozliczeniowych dla danych MQI.
Statystyki MQIStatistics	Steruje gromadzeniem informacji o monitorowaniu statystyki dla menedżera kolejek.
MsgMarkBrowseInterval	Odstęp czasu, po upływie którego menedżer kolejek może usunąć znacznik z przeglądanych komunikatów.
OutboundPortMin	W przypadku parametru *OutboundPortMin*definiuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.

Tabela 557. Atrybuty menedżera kolejek (kontynuacja)	
Atrybut	**Opis**
OutboundPortMin	W przypadku parametru *OutboundPortMax*definiuje zakres numerów portów, które mają być używane podczas wiązania kanałów wychodzących.
PerformanceEvent	Określa, czy generowane są zdarzenia związane z wydajnością
Platforma	Platforma, na której jest uruchomiony menedżer kolejek
PubSubNPInputMsg	Czy odrzucić (lub zachować) niedostarczony komunikat wejściowy
PubSubNPResponse (odpowiedź NPResponse)	Steruje zachowaniem niedostarczonych
PubSubMaxMsgRetryCount	Liczba ponowień podczas przetwarzania (w punkcie synchronizacji) komunikatu komendy zakończonej niepowodzeniem
PubSubSyncPoint	Określa, czy w punkcie synchronizacji mają być przetwarzane tylko trwałe (lub wszystkie) komunikaty
TrybPubSub	Określa, czy działa umieszczony w kolejce interfejs publikowania/subskrybowania
QMgrDesc	Opis menedżera kolejek
QMgrIdentifier	Unikalny, wygenerowany wewnętrznie identyfikator menedżera kolejek
QMgrName	Nazwa menedżera kolejek
Nazwa QSGName	Nazwa grupy współużytkowania kolejek
QueueAccounting	Steruje kolekcjonowaniem informacji rozliczeniowych dla kolejek.
QueueMonitoring	Dane monitorowania bezpośredniego dla kolejek
QueueStatistics	Steruje gromadzeniem danych statystycznych dla kolejek.
ReceiveTimeout	Czas oczekiwania kanału TCP/IP na dane przed powrotem do stanu nieaktywnego.
ReceiveTimeoutMin	Kwalifikator dla *ReceiveTimeout*.
TypReceiveTimeout	Minimalny czas oczekiwania kanału TCP/IP na dane przed powrotem do stanu nieaktywnego.
RemoteEvent	Określa, czy generowane są zdalne zdarzenia błędów
RepositoryName	Nazwa klastra, dla którego ten menedżer kolejek udostępnia usługi repozytorium
RepositoryNamelist	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługi repozytorium
ScyCase	Przypadek profili zabezpieczeń
SharedQMgrNazwa	Nazwa menedżera kolejek współużytkowanych
“SPLCAP” na stronie 859	IBM MQ Zaawansowana ochrona zabezpieczeń komunikatów dla menedżera kolejek jest włączona lub wyłączona.
Lista nazw SSL [1]	Nazwa obiektu listy nazw zawierającego nazwy obiektów informacji uwierzytelniającej.
SSLCryptoHardware [1]	Łańcuch konfiguracji sprzętu szyfrującego.
odpowietrznik (SSLEvent)	Atrybut sterujący dla zdarzeń TLS.
SSLFIPSREQUIRED	Do szyfrowania należy używać tylko algorytmów z certyfikatem FIPS.
SSLKeyRepository [1]	Położenie repozytorium kluczy TLS.
V 9.3.0 V 9.3.0 SSLKeyRepositoryHasło [1]	Hasło do repozytorium kluczy TLS.
Liczba:SSLKeyReset	Licznik resetowania klucza TLS.
Zadania SSL [1]	Liczba podzadań serwera do przetwarzania wywołań TLS.
StatisticsInterval	Częstotliwość zapisywania danych monitorowania statystyk.
ZdarzenieStartStop	Określa, czy generowane są zdarzenia uruchomienia i zatrzymania
SyncPoint	Dostępność punktu synchronizacji
Kanały TCP	Maksymalna liczba bieżących kanałów lub podłączonych klientów używających protokołu TCP/IP.
TCPKeepAlive	Określa, czy użyć TCP KEEPALIVE do sprawdzenia innego końca połączenia.
TCPNAME	Nazwa używanego systemu TCP/IP.
TCPStackType	Sposób, w jaki inicjator kanału może używać adresów TCP/IP.

Tabela 557. Atrybuty menedżera kolejek (kontynuacja)	
Atrybut	**Opis**
TraceRouteRejestrowanie, atrybut	Steruje rejestrowaniem informacji o trasie śledzenia.
TriggerInterval	Odstęp czasu między komunikatami wyzwalacza
Wersja	Wersja
XrCapability	Określa, czy obsługiwane są komendy składnika Telemetry.
Uwagi: 1. Ten atrybut nie może zostać sprawdzony przy użyciu wywołania MQINQ i nie jest opisany w tej sekcji. Szczegółowe informacje na temat tego atrybutu zawiera sekcja Zmiana menedżera kolejek .	

Zadania pokrewne

Określenie, że w czasie wykonywania na kliencie MQI będą używane tylko CipherSpecs z certyfikatem FIPS.

Odsyłacze pokrewne

Standardy FIPS (Federal Information Processing Standards) dla AIX, Linux, and Windows

AccountingConnNadpisanie (MQLONG)

Dzięki temu aplikacje mogą przesłonić ustawienia wartości ACCTMQI i ACCTQDATA w atrybucie Qmgr.

Jest to jedna z następujących wartości:

MQMON_DISABLED (wyłączone)
Aplikacje nie mogą przesłonić ustawienia atrybutów ACCTMQI i ACCTQ Qmgr za pomocą pola Opcje w strukturze MQCNO wywołania MQCONNX. Jest to wartość domyślna.

MQMON_ENABLED
Aplikacje mogą przesłonić atrybuty ACCTQ i ACCTMQI Qmgr za pomocą pola Opcje w strukturze MQCNO.

Zmiany tej wartości mają zastosowanie tylko w przypadku połączeń z menedżerem kolejek po zmianie atrybutu.

Ten atrybut jest obsługiwany tylko na następujących platformach:

- IBM i
- AIX and Linux
- Windows

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_CONN_OVERRIDE z wywołaniem MQINQ.

AccountingInterval (MQLONG)

Określa czas przed zapisaniem pośrednich rekordów rozliczeniowych (w sekundach).

Wartością jest liczba całkowita z zakresu od 0 do 604800, a wartością domyślną jest 1800 (30 minut). Podaj 0, aby wyłączyć rekordy pośrednie.

Ten atrybut jest obsługiwany tylko na następujących platformach:

- IBM i
- AIX and Linux
- Linux
- Windows

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_INTERVAL z wywołaniem MQINQ.

Przesłonięcie ActivityConn(MQLONG)

Dzięki temu aplikacje mogą przesłonić ustawienie wartości ACTVTRC w atrybucie menedżera kolejek.

Jest to jedna z następujących wartości:

MQMON_DISABLED (wyłączone)
Aplikacje nie mogą przesłonić ustawienia atrybutu menedżera kolejek ACTVTRC za pomocą pola Opcje w strukturze MQCNO wywołania MQCONNX. Jest to wartość domyślna.

MQMON_ENABLED
Aplikacje mogą przesłonić atrybut menedżera kolejek ACTVTRC za pomocą pola Opcje w strukturze MQCNO.

Zmiany tej wartości mają zastosowanie tylko w przypadku połączeń z menedżerem kolejek po zmianie atrybutu.

Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACTIVITY_CONN_OVERRIDE z wywołaniem MQINQ .

ActivityTrace (MQLONG)

Steruje to gromadzeniem danych śledzenia działań aplikacji MQI produktu IBM MQ .

Jest to jedna z następujących wartości:

MQMON_ON
Zgromadź dane śledzenia działań aplikacji MQI produktu IBM MQ .

MQMON_WYŁ
Nie należy gromadzić danych śledzenia działań aplikacji MQI produktu IBM MQ . Jest to wartość domyślna.

Jeśli atrybut menedżera kolejek ACTVCONO zostanie ustawiony na wartość ENABLED, ta wartość może zostać nadpisana dla pojedynczych połączeń przy użyciu pola Opcje w strukturze MQCNO.

Zmiany tej wartości mają zastosowanie tylko w przypadku połączeń z menedżerem kolejek po zmianie atrybutu.

Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACTIVITY_TRACE z wywołaniem MQINQ .

AdoptNewMCACheck (MQLONG)

Definiuje elementy, które mają zostać sprawdzone w celu określenia, czy należy zastosować agent MCA w przypadku wykrycia nowego kanału przychodzącego, który ma taką samą nazwę jak agent MCA, który jest już aktywny.

Jest to jedna z następujących wartości:

MQADOPT_CHECK_Q_MGR_NAME
Sprawdź nazwę menedżera kolejek.

MQADOPT_CHECK_NET_ADDR
Sprawdź adres sieciowy.

MQADOPT_CHECK_ALL
Sprawdź nazwę menedżera kolejek i adres sieciowy. Jeśli to możliwe, należy wykonać to sprawdzenie, aby zabezpieczyć kanały przed zamknięciem, nieumyślnym lub złośliwym zamknięciem. Jest to wartość domyślna.

MQADOPT_CHECK_NONE
Nie sprawdzaj żadnych elementów.

Zmiany tego atrybutu są uwzględniane przy następnej próbie zaadoptowania kanału przez kanał.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ADOPTNEWMCA_CHECK z wywołaniem MQINQ.

AdoptNewMCAType (MQLONG)

Określa, czy automatycznie restartować osieroconą instancję agenta MCA określonego typu kanału w przypadku wykrycia nowego żądania kanału przychodzącego zgodnego z atrybutem MCACheck AdoptNew.

Jest to jedna z następujących wartości:

MQADOPT_TYPE_NO
Adoptowanie osieroconych instancji kanału nie jest wymagane. Jest to wartość domyślna.

MQADOPT_TYPE_ALL
Adoptuj wszystkie typy kanałów.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ADOPTNEWMCA_TYPE z wywołaniem MQINQ.

AlterationDate (MQCHAR12)

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_TIME_LENGTH.

AuthorityEvent (MQLONG)

Ta opcja określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane). Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_AUTHORITY_EVENT z wywołaniem MQINQ.

BridgeEvent (MQLONG)

Określa, czy generowane są zdarzenia mostu IMS .

Jest to jedna z następujących wartości:

MQEVR_ENABLED,
Wygeneruj zdarzenia mostu IMS w następujący sposób:

MQRC_BRIDGE_STARTED,
MQRC_BRIDGE_STOPPED

MQEVR_DISABLED (wyłączone)
Nie generuj zdarzeń mostu IMS . Jest to wartość domyślna.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_BRIDGE_EVENT z wywołaniem MQINQ.

ChannelAutoDef (MQLONG)

Ten atrybut steruje automatyczną definicją kanałów typu MQCHT_RECEIVER i MQCHT_SVRCONN. Automatyczna definicja kanałów MQCHT_CLUSSDR jest zawsze włączona. Jest to jedna z następujących wartości:

MQCHAD_WYŁĄCZONE
Automatyczne definiowanie kanału jest wyłączone.

MQCHAD_ENABLED
Automatyczne definiowanie kanału jest włączone.

Multi Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHANNEL_AUTO_DEF z wywołaniem MQINQ.

ChannelAutoDefEvent (MQLONG)

Określa, czy są generowane zdarzenia automatycznej definicji kanału. Dotyczy to kanałów typu MQCHT_RECEIVER, MQCHT_SVRCONN i MQCHT_CLUSSDR. Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Multi Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHANNEL_AUTO_DEF_EVENT z wywołaniem MQINQ.

ChannelAutoDefExit (MQCHARn)

Jest to nazwa procedury zewnętrznej dla automatycznej definicji kanału. Jeśli ta nazwa nie jest pusta, a parametr *ChannelAutoDef* ma wartość MQCHAD_ENABLED, wyjście jest wywoływane za każdym razem, gdy menedżer kolejek ma zamiar utworzyć definicję kanału. Dotyczy to kanałów typu MQCHT_RECEIVER, MQCHT_SVRCONN i MQCHT_CLUSSDR. Wyjście może następnie wykonać jedną z następujących czynności:

- Utwórz definicję kanału bez zmian.
- Zmodyfikuj atrybuty tworzonej definicji kanału.
- Całkowicie pomijaj tworzenie kanału.

Uwaga: Zarówno długość, jak i wartość tego atrybutu są specyficzne dla środowiska. Szczegółowe informacje na temat wartości tego atrybutu w różnych środowiskach można znaleźć we wprowadzeniu do struktury MQCD w sekcji “MQCD-definicja kanału” na stronie 1526 .

z/OS W systemie z/OSten atrybut dotyczy tylko kanałów nadawczych i odbiorczych klastra.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CHANNEL_AUTO_DEF_EXIT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_EXIT_NAME_LENGTH.

ChannelEvent (MQLONG)
Określa, czy generowane są zdarzenia kanału.

Jest to jedna z następujących wartości:

MQEVR_EXCEPTION,
Generuj tylko następujące zdarzenia kanału:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED z następującymi ReasonQualifiers:

  MQRQ_CHANNEL_STOPPED_ERROR
  MQRQ_CHANNEL_STOPPED_RETRY
  MQRQ_CHANNEL_STOPPED_DISABLED

  MQRC_CHANNEL_STOPPED_BY_USER

MQEVR_ENABLED,
Generuj wszystkie zdarzenia kanału. Oznacza to, że oprócz zdarzeń wygenerowanych przez parametr EXCEPTION, generowane są następujące zdarzenia kanału:

- MQRC_CHANNEL_STARTED
- MQRC_CHANNEL_STOPPED z następującym atrybutem ReasonQualifier:

  MQRQ_CHANNEL_STOPPED_OK

MQEVR_DISABLED (wyłączone)
Nie generuj zdarzeń kanału. Jest to wartość domyślna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHANNEL_EVENT z wywołaniem MQINQ.

Element sterujący ChannelInitiator(MQLONG)
Określa, czy inicjator kanału ma być uruchamiany podczas uruchamiania menedżera kolejek.

Jest to jedna z następujących wartości:

MQSVC_CONTROL_MANUAL,
Inicjator kanału nie ma być uruchamiany automatycznie.

MQSVC_CONTROL_Q_MGR
Inicjator kanału ma być uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_CONTROL z wywołaniem MQINQ.

ChannelMonitoring (MQLONG)
Ten atrybut określa dane monitorowania bezpośredniego dla kanałów.

Jest to jedna z następujących wartości:

MQMON_BRAK
Należy wyłączyć gromadzenie danych dla monitorowania wszystkich kanałów bez względu na ustawienie atrybutu kanału MONCHL. Jest to wartość domyślna.

MQMON_WYŁ
Wyłącz gromadzenie danych monitorowania dla kanałów, które określają QMGR w atrybucie kanału MONCHL.

MQMON_NISKI
Kolekcjonowanie danych monitorowania należy włączyć przy niskim współczynniku kolekcjonowania danych dla kanałów, dla których określono QMGR w atrybucie kanału MONCHL.

MQMON_MEDIUM
Kolekcjonowanie danych monitorowania należy włączyć z umiarkowanym współczynnikiem kolekcjonowania danych dla kanałów, w których określono QMGR w atrybucie kanału MONCHL.

MQMON_HIGH
Włącz kolekcjonowanie danych monitorowania z wysokim współczynnikiem kolekcjonowania danych dla kanałów, dla których określono QMGR w atrybucie kanału MONCHL.

z/OS W systemach z/OS włączenie tego parametru powoduje po prostu włączenie gromadzenia danych statystycznych, niezależnie od wybranej wartości. Ustawienie opcji LOW, MEDIUM lub HIGH nie ma wpływu na wyniki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_CHANNEL z wywołaniem MQINQ.

ChannelStatistics (MQLONG)

Steruje to gromadzeniem danych statystycznych dla kanałów.

Jest to jedna z następujących wartości:

MQMON_BRAK
Wyłącz gromadzenie danych dla statystyk wszystkich kanałów bez względu na ustawienie atrybutu kanału STATCHL. Jest to wartość domyślna.

MQMON_WYŁ
Wyłącz gromadzenie danych statystycznych dla kanałów, które określają QMGR w atrybucie kanału STATCHL.

MQMON_NISKI
Należy włączyć gromadzenie danych statystycznych przy niskim współczynniku gromadzenia danych dla kanałów, dla których określono QMGR w atrybucie kanału STATCHL.

MQMON_MEDIUM
Włącz kolekcjonowanie danych statystycznych z umiarkowanym współczynnikiem kolekcjonowania danych dla kanałów, dla których określono QMGR w atrybucie kanału STATCHL.

MQMON_HIGH
Włącz gromadzenie danych statystycznych z wysokim współczynnikiem gromadzenia danych dla kanałów, dla których określono QMGR w atrybucie kanału STATCHL.

W większości systemów zaleca się używanie MEDIUM. Jednak w przypadku kanału, który przetwarza dużą liczbę komunikatów na sekundę, można zmniejszyć poziom próbkowania, wybierając opcję LOW. Ponadto w przypadku kanału, który przetwarza tylko kilka komunikatów i dla którego najważniejsze są najbardziej aktualne informacje, można wybrać opcję HIGH.

z/OS W systemach z/OS włączenie tego parametru powoduje po prostu włączenie gromadzenia danych statystycznych, niezależnie od wybranej wartości. Ustawienie opcji LOW, MEDIUM lub HIGH nie ma wpływu na wyniki.Ten parametr musi być włączony, aby były gromadzone rekordy rozliczeniowe kanałów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_CHANNEL z wywołaniem MQINQ.

ChinitAdapters (MQLONG)

Jest to liczba podzadań adaptera, które mają być używane do przetwarzania wywołań IBM MQ . Wartość musi być z zakresu od 0 do 9999, a wartością domyślną jest 8.

Stosunek liczby adapterów do liczby programów rozsyłających (atrybut ChinitDispatchers) powinien wynosić od 8 do 5. Jeśli jednak istnieje tylko kilka kanałów, nie trzeba zmniejszać wartości tego parametru z wartości domyślnej. Można użyć następujących wartości: dla systemu testowego, 8 (wartość domyślna); dla systemu produkcyjnego, 20. W idealnym przypadku powinno być 20 adapterów, co zapewnia większy paralelizm wywołań IBM MQ . Jest to istotne w przypadku komunikatów trwałych. Mniejsza liczba adapterów może być lepsza w przypadku komunikatów nietrwałych.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_ADAPTERS z wywołaniem MQINQ.

ChinitDispatchers (MQLONG)

Jest to liczba programów rozsyłających, które mają być używane dla inicjatora kanału. Wartość musi być z zakresu od 0 do 9999, a wartością domyślną jest 5.

Jako wskazówkę należy zezwolić na jeden przekaźnik dla 50 bieżących kanałów. Jeśli jednak istnieje tylko kilka kanałów, nie trzeba zmniejszać wartości tego atrybutu z wartości domyślnej. Jeśli używany jest protokół TCP/IP, największa liczba programów rozsyłających, które są używane dla kanałów TCP/IP, wynosi 100, nawet jeśli określono w tym miejscu większą wartość. Można użyć następujących ustawień: systemy testowe, 5 (wartość domyślna), systemy produkcyjne, 20 (do obsługi 1000 aktywnych kanałów potrzeba 20 przekaźników).

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_DISPATCHERS z wywołaniem MQINQ.

ChinitTraceAutoStart (MQLONG)

Określa, czy śledzenie inicjatora kanału ma być uruchamiane automatycznie.

Jest to jedna z następujących wartości:

MQTRAXSTR_YES
Automatycznie uruchom śledzenie inicjatora kanału. Jest to wartość domyślna.

MQTRAXSTR_NO
Nie uruchamiaj automatycznie śledzenia inicjatora kanału.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_TRACE_AUTO_START z wywołaniem MQINQ.

ChinitTraceTableSize (MQLONG)

Jest to wielkość obszaru danych śledzenia inicjatora kanału (w MB).

Wartość musi należeć do zakresu od 0 do 2048, a wartością domyślną jest 2.

Uwaga: Za każdym razem, gdy używane są duże obszary danych z/OS , należy upewnić się, że w systemie jest wystarczająca ilość pamięci dyskowej do obsługi działań stronicowania związanego z systemem z/OS . Może także być konieczne zwiększenie wielkości zestawów danych SYS1.DUMP.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CHINIT_TRACE_TABLE_SIZE z wywołaniem MQINQ.

ClusterSenderMonitoringDefault (MQLONG)

Określa wartość, która ma zostać zastąpiona atrybutem ChannelMonitoring automatycznie definiowanych kanałów nadawczych klastra.

Jest to jedna z następujących wartości:

MQMON_Q_MGR
Kolekcja danych monitorowania bezpośredniego jest dziedziczona z ustawienia atrybutu **ChannelMonitoring** menedżera kolejek. Jest to wartość domyślna.

MQMON_WYŁ
Monitorowanie kanału jest wyłączone

MQMON_NISKI
Jeśli parametr *ChannelMonitoring* nie ma wartości MQMON_NONE, monitorowanie jest włączone z małą szybkością gromadzenia danych przy minimalnym wpływie na wydajność systemu. Zgromadzone dane prawdopodobnie nie są najbardziej aktualne.

MQMON_MEDIUM
Jeśli parametr *ChannelMonitoring* nie ma wartości MQMON_NONE, monitorowanie jest włączone z umiarkowaną szybkością gromadzenia danych, co ma ograniczony wpływ na wydajność systemu.

MQMON_HIGH
Jeśli parametr *ChannelMonitoring* nie ma wartości MQMON_NONE, monitorowanie jest włączone z dużą szybkością gromadzenia danych, co może mieć wpływ na wydajność systemu. Zebrane dane są najbardziej aktualne.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_AUTO_CLUSSDR z wywołaniem MQINQ.

Statystyki ClusterSender(MQLONG)

Ponieważ kanały nadawcze klastra mogą być automatycznie definiowane na podstawie definicji CLUSRCVR w repozytorium, nie można zmienić ustawienia atrybutu STATCHL dla tych automatycznie definiowanych kanałów nadawczych klastra za pomocą instrukcji ALTER channel. W przypadku tych kanałów decyzja o tym, czy kolekcjonować dane monitorowania bezpośredniego, jest oparta na ustawieniu tego atrybutu menedżera kolejek.

Jest to jedna z następujących wartości:

MQMON_Q_MGR
Gromadzenie danych statystycznych dla automatycznie definiowanych kanałów wysyłających klastry jest oparte na wartości atrybutu STATCHL menedżera kolejek. Jest to wartość domyślna.

MQMON_WYŁ
Wyłącz gromadzenie danych statystycznych dla automatycznie definiowanych kanałów wysyłających klastry.

MQMON_NISKI
Włącz gromadzenie danych statystycznych dla automatycznie definiowanych kanałów wysyłających klastry z niskim współczynnikiem gromadzenia danych.

MQMON_MEDIUM
Włącz gromadzenie danych statystycznych dla automatycznie definiowanych kanałów wysyłających klastry z umiarkowanym współczynnikiem gromadzenia danych.

MQMON_HIGH
Włącz gromadzenie danych statystycznych dla automatycznie definiowanych kanałów nadawczych klastra o wysokim współczynniku gromadzenia danych.

Dla większości systemów zalecamy MEDIUM. Jednak w przypadku automatycznie definiowanego kanału nadawczego klastra, który przetwarza dużą liczbę komunikatów na sekundę, można zmniejszyć poziom próbkowania, wybierając opcję LOW. Ponadto w przypadku kanału, który przetwarza tylko kilka komunikatów i dla którego najważniejsze są najbardziej aktualne informacje, można wybrać opcję HIGH.

z/OS W systemach z/OS włączenie tego parametru powoduje po prostu włączenie gromadzenia danych statystycznych, niezależnie od wybranej wartości. Ustawienie opcji `LOW`, `MEDIUM` lub `HIGH` nie ma wpływu na wyniki.Ten parametr musi być włączony, aby były gromadzone rekordy rozliczeniowe kanałów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_AUTO_CLUSSDR z wywołaniem MQINQ.

ClusterWorkloadDane (MQCHAR32)

Jest to zdefiniowany przez użytkownika 32-bajtowy łańcuch znaków, który jest przekazywany do wyjścia obciążenia klastra w momencie jego wywołania. Jeśli nie ma danych do przekazania do wyjścia, łańcuch jest pusty.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLUSTER_WORKLOAD_DATA z wywołaniem MQINQ.

Wyjście ClusterWorkload(MQCHARn)

Jest to nazwa procedury zewnętrznej dla zarządzania obciążeniem klastra. Jeśli ta nazwa nie jest pusta, wyjście jest wywoływane za każdym razem, gdy komunikat jest umieszczany w kolejce klastra lub przenoszony z jednej kolejki nadawczej klastra do innej. Wyjście może następnie zaakceptować instancję kolejki wybraną przez menedżera kolejek jako miejsce docelowe komunikatu lub wybrać inną instancję kolejki.

Uwaga: Zarówno długość, jak i wartość tego atrybutu są specyficzne dla środowiska.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLUSTER_WORKLOAD_EXIT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_EXIT_NAME_LENGTH.

Długość ClusterWorkload(MQLONG)

Jest to maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra. Rzeczywista długość danych przekazywanych do wyjścia wynosi co najmniej:

- Długość komunikatu.
- Atrybut **MaxMsgLength** menedżera kolejek.
- Atrybut **ClusterWorkloadLength** .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLUSTER_WORKLOAD_LENGTH z wywołaniem MQINQ.

Kanały CLWLMRUChannel (MQLONG)

Określa maksymalną liczbę ostatnio używanych kanałów klastra, które mają być używane przez algorytm wyboru obciążenia klastra.

Jest to wartość z zakresu od 1 do 999999999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_MRU_CHANNELS z wywołaniem MQINQ.

CLWLUseQ (MQLONG),

Określa, czy dla obciążenia klastra mają być używane kolejki zdalne.

Jest to jedna z następujących wartości:

MQCLWL_USEQ_ANY
Użyj kolejek lokalnych i zdalnych.

MQCLWL_USEQ_LOCAL.
Nie należy używać kolejek zdalnych. Jest to wartość domyślna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_USEQ z wywołaniem MQINQ.

CodedCharSetId (MQLONG)

Definiuje zestaw znaków używany przez menedżera kolejek dla wszystkich pól łańcucha znaków zdefiniowanych w MQI, takich jak nazwy obiektów oraz data i godzina utworzenia kolejki. Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach obiektów. Nie ma ono zastosowania do danych aplikacji zawartych w komunikacie. Wartość zależy od środowiska:

- W systemie z/OSwartość jest ustawiana na podstawie parametrów systemowych podczas uruchamiania menedżera kolejek. Wartość domyślna to 500.
- W systemie Windowswartością jest podstawowa strona kodowa (CODEPAGE) użytkownika tworzącego menedżer kolejek.
- W systemie IBM iwartością jest wartość ustawiona w środowisku, gdy menedżer kolejek jest tworzony po raz pierwszy.

- W systemie AIX and Linuxwartością jest domyślny CODESET dla ustawień narodowych użytkownika tworzącego menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CODED_CHAR_SET_ID z wywołaniem MQINQ.

CommandEvent (MQLONG)

Określa, czy zdarzenia komendy są generowane w następujący sposób:

MQEVR_DISABLED (wyłączone)
Nie generuj zdarzeń komend. Jest to opcja domyślna.

MQEVR_ENABLED,
Wygeneruj zdarzenia komendy.

MQEVR_NO_DISPLAY
Zdarzenia komend są generowane dla wszystkich pomyślnych komend innych niż MQINQ.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_COMMAND_EVENT z wywołaniem MQINQ.

CommandInputQName (MQCHAR48)

Jest to nazwa kolejki wejściowej komend zdefiniowanej w menedżerze kolejek lokalnych. Jest to kolejka, do której użytkownicy mogą wysyłać komendy, jeśli są do tego uprawnieni. Nazwa kolejki zależy od środowiska:

- W systemie z/OSnazwa kolejki to SYSTEM.COMMAND.INPUT; można do niego wysyłać komendy MQSC i PCF. Szczegółowe informacje na temat komend PCF zawiera sekcja Komendy MQSC oraz sekcja Definicje formatów komend programowalnych .
- We wszystkich innych środowiskach nazwa kolejki to SYSTEM.ADMIN.COMMAND.QUEUEi można do niej wysyłać tylko komendy PCF. Jednak komenda MQSC może zostać wysłana do tej kolejki, jeśli komenda MQSC jest ujęta w komendzie PCF typu MQCMD_ESCAPE. Więcej informacji na temat komendy Escape zawiera sekcja Escape .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_COMMAND_INPUT_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

CommandLevel (MQLONG)

Uwaga: Obsługa systemu operacyjnego HP-UX dla wszystkich komponentów IBM MQ , w tym serwera i klientów, została usunięta w produkcie IBM MQ 9.1.

Wskazuje poziom komend sterowania systemem obsługiwanych przez menedżer kolejek. Może to być jedna z następujących wartości:

MQCMDL_LEVEL_800
Poziom 800 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 8.0
- IBM MQ for IBM i 8.0
- IBM MQ for Linux 8.0
- IBM MQ for Windows 8.0
- IBM MQ for z/OS 8.0

MQCMDL_LEVEL_801
Poziom 801 systemowych komend sterujących.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 8.0.0 Fix Pack 2
- IBM MQ for HP-UX 8.0.0 Fix Pack 2
- IBM MQ for IBM i 8.0.0 Fix Pack 2
- IBM MQ for Linux 8.0.0 Fix Pack 2

MQCMDL_LEVEL_802
Poziom 802 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 8.0.0 Fix Pack 3
- IBM MQ for IBM i 8.0.0 Fix Pack 3
- IBM MQ for Linux 8.0.0 Fix Pack 3
- IBM MQ for Windows 8.0.0 Fix Pack 3

MQCMDL_LEVEL_900
Poziom 900 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.0
- IBM MQ for IBM i 9.0
- IBM MQ for Linux 9.0
- IBM MQ for Windows 9.0
- IBM MQ for z/OS 9.0

MQCMDL_LEVEL_901
Poziom 901 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for Linux 9.0.1
- IBM MQ for Windows 9.0.1
- IBM MQ for z/OS 9.0.1

MQCMDL_LEVEL_902
Poziom 902 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for Linux 9.0.2
- IBM MQ for Windows 9.0.2
- IBM MQ for z/OS 9.0.2

MQCMDL_LEVEL_903
Poziom 903 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for Linux 9.0.3
- IBM MQ for Windows 9.0.3
- IBM MQ for z/OS 9.0.3

MQCMDL_LEVEL_904
Poziom 904 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.0.4
- IBM MQ for Linux 9.0.4
- IBM MQ for Windows 9.0.4

- IBM MQ for z/OS 9.0.4

MQCMDL_LEVEL_905
Poziom 905 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.0.5
- IBM MQ for Linux 9.0.5
- IBM MQ for Windows 9.0.5
- IBM MQ for z/OS 9.0.5

MQCMDL_LEVEL_910
Poziom 910 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_911
Poziom 911 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1.1
- IBM MQ for Linux 9.1.1
- IBM MQ for Windows 9.1.1
- IBM MQ for z/OS 9.1.1

MQCMDL_LEVEL_912
Poziom 912 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1.2
- IBM MQ for Linux 9.1.2
- IBM MQ for Windows 9.1.2
- IBM MQ for z/OS 9.1.2

MQCMDL_LEVEL_913
Poziom 913 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1.3
- IBM MQ for Linux 9.1.3
- IBM MQ for Windows 9.1.3
- IBM MQ for z/OS 9.1.3

MQCMDL_LEVEL_914
Poziom 914 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1.4
- IBM MQ for Linux 9.1.4
- IBM MQ for Windows 9.1.4

- IBM MQ for z/OS 9.1.4

MQCMDL_LEVEL_915
Poziom 915 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1.5
- IBM MQ for Linux 9.1.5
- IBM MQ for Windows 9.1.5
- IBM MQ for z/OS 9.1.5

MQCMDL_LEVEL_910
Poziom 910 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.1
- IBM MQ for IBM i 9.1
- IBM MQ for Linux 9.1
- IBM MQ for Windows 9.1
- IBM MQ for z/OS 9.1

MQCMDL_LEVEL_920
Poziom 920 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.2
- IBM MQ for IBM i 9.2
- IBM MQ for Linux 9.2
- IBM MQ for Windows 9.2
- IBM MQ for z/OS 9.2

MQCMDL_LEVEL_921
Poziom 921 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.2.1
- IBM MQ for Linux 9.2.1
- IBM MQ for Windows 9.2.1
- IBM MQ for z/OS 9.2.1

MQCMDL_LEVEL_922
Poziom 922 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.2.2
- IBM MQ for Linux 9.2.2
- IBM MQ for Windows 9.2.2
- IBM MQ for z/OS 9.2.2

MQCMDL_LEVEL_923
Poziom 923 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.2.3
- IBM MQ for Linux 9.2.3

- IBM MQ for Windows 9.2.3
- IBM MQ for z/OS 9.2.3

MQCMDL_LEVEL_924
Poziom 924 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.2.4
- IBM MQ for Linux 9.2.4
- IBM MQ for Windows 9.2.4
- IBM MQ for z/OS 9.2.4

MQCMDL_LEVEL_925
Poziom 925 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.2.5
- IBM MQ for Linux 9.2.5
- IBM MQ for Windows 9.2.5
- IBM MQ for z/OS 9.2.5

MQCMDL_LEVEL_930
Poziom 930 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.3
- IBM MQ for IBM i 9.3
- IBM MQ for Linux 9.3
- IBM MQ for Windows 9.3
- IBM MQ for z/OS 9.3

MQCMDL_LEVEL_931
Poziom 931 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.3.1
- IBM MQ for Linux 9.3.1
- IBM MQ for Windows 9.3.1
- IBM MQ for z/OS 9.3.1

MQCMDL_LEVEL_932
Poziom 932 komend sterowania systemem.

Ta wartość jest zwracana przez następujące wersje:

- IBM MQ for AIX 9.3.2
- IBM MQ for Linux 9.3.2
- IBM MQ for Windows 9.3.2
- IBM MQ for z/OS 9.3.2

Zestaw komend sterujących systemu, który odpowiada konkretnej wartości atrybutu **CommandLevel** , różni się w zależności od wartości atrybutu **Platform** . Obie te komendy muszą zostać użyte do określenia, które z nich są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_COMMAND_LEVEL z wywołaniem MQINQ .

Element sterujący CommandServer(MQLONG)

Określa, czy serwer komend ma być uruchamiany podczas uruchamiania menedżera kolejek.

Możliwe wartości:

MQSVC_CONTROL_MANUAL,
Serwer komend nie ma być uruchamiany automatycznie.

MQSVC_CONTROL_Q_MGR
Serwer komend ma być uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

Ten atrybut nie jest obsługiwany w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CMD_SERVER_CONTROL z wywołaniem MQINQ.

ConfigurationEvent (MQLONG)

Określa, czy generowane są zdarzenia konfiguracji.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CONFIGURATION_EVENT z wywołaniem MQINQ.

Możliwe wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Multi *CurrentQFile(MQLONG)*

Bieżąca wielkość pliku kolejki w megabajtach, zaokrąglona w górę do najbliższego megabajta.

Tabela 558. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Wartość tego atrybutu statusu kolejki jest równa bieżącej wielkości kolejki, zaokrąglonej w górę do najbliższego megabajta. Dla nowej kolejki z atrybutami domyślnymi **CurrentQFileSize** ma wartość 1.

Maksymalna wartość tego atrybutu wynosi 99,999,9999 MB, a dla tego atrybutu nie ma wartości domyślnej.

Multi *CurrentMaxQFileSize (MQLONG)*

Bieżąca maksymalna wielkość, do której może zostać powiększony plik kolejki, zaokrąglona w górę do najbliższego megabajta, biorąc pod uwagę bieżącą wielkość bloku, która jest używana w kolejce.

Tabela 559. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Pole to składa się z dwóch części:

- Jeśli parametr **MaxQFileSize** zostanie ustawiony na wartość domyślną dla bieżącej wielkości bloku, program **CurrentMaxQFileSize** wyświetli rzeczywistą wartość, której odpowiada wartość domyślna.
- Jeśli parametr **CurrentMaxQFileSize** nie jest zgodny z wartością **MaxQFileSize**, oznacza to, że kolejka musi zostać wyczerpana, aby możliwe było przyjęcie większej granulacji.

Uwaga: Więcej informacji na temat zmiany wielkości plików kolejki oraz wielkości bloku i granulacji zawiera sekcja Modyfikowanie plików kolejki produktu IBM MQ .

Maksymalna wartość tego atrybutu wynosi 99,999,9999 MB i nie ma wartości domyślnej. Wartość ta jest wartością maksymalną, która jest obecnie ustawiona; dla nowej kolejki z atrybutami domyślnymi, wartość **CurrentMaxQFileSize** wynosi 2,088,960 MB.

DeadLetterQName (MQCHAR48)

Jest to nazwa kolejki zdefiniowana w menedżerze kolejek lokalnych jako kolejka niedostarczonych komunikatów. Komunikaty są wysyłane do tej kolejki, jeśli nie można ich skierować do właściwego miejsca docelowego.

Na przykład komunikaty są umieszczane w tej kolejce, gdy:

- Komunikat dociera do menedżera kolejek i jest przeznaczony dla kolejki, która nie została jeszcze zdefiniowana w tym menedżerze kolejek.
- Komunikat dociera do menedżera kolejek, ale kolejka, do której jest przeznaczony, nie może go odebrać, ponieważ prawdopodobnie:
  - Kolejka jest pełna
  - Żądania umieszczania są zablokowane
  - Węzeł wysyłający nie ma uprawnień do umieszczania komunikatów w kolejce

Aplikacje mogą również umieszczać komunikaty w kolejce niedostarczonych komunikatów.

Komunikaty raportu są traktowane w taki sam sposób, jak zwykłe komunikaty. Jeśli komunikat raportu nie może zostać dostarczony do kolejki docelowej (zwykle jest to kolejka określona w polu *ReplyToQ* w deskryptorze oryginalnego komunikatu), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów.

Uwaga: Komunikaty, dla których upłynął czas ważności (patrz MQMD-pole Utrata ważności) **nie** są przesyłane do tej kolejki, gdy są usuwane. Jednak komunikat raportu o utracie ważności (MQRO_EXPIRATION) jest nadal generowany i wysyłany do kolejki *ReplyToQ* , jeśli zostanie zażądany przez aplikację wysyłającą.

Komunikaty nie są umieszczane w kolejce niedostarczonych komunikatów, jeśli aplikacja, która wysłała żądanie umieszczenia, została powiadomiona synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 (na przykład komunikat umieszczony w kolejce lokalnej, dla którego żądania umieszczenia są zablokowane).

Dane komunikatów aplikacji w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) mają czasami przedrostek w postaci struktury MQDLH. Ta struktura zawiera dodatkowe informacje wskazujące, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Więcej informacji na temat tej struktury zawiera sekcja “MQDLH-nagłówek niedostarczonego komunikatu” na stronie 359 .

Ta kolejka musi być kolejką lokalną z atrybutem **Usage** o wartości MQUS_NORMAL.

Jeśli menedżer kolejek nie obsługuje kolejki niedostarczonych komunikatów (niedostarczonych komunikatów) lub jeśli kolejka nie została zdefiniowana, nazwa jest pusta. Wszystkie menedżery kolejek produktu IBM MQ obsługują kolejkę niedostarczonych komunikatów (niedostarczonych komunikatów), ale domyślnie nie jest ona zdefiniowana.

Jeśli kolejka niedostarczonych komunikatów (niedostarczonych komunikatów) nie jest zdefiniowana, pełna lub nieużyteczna z innego powodu, komunikat, który zostałby do niej przesłany przez agenta kanału komunikatów, jest zachowywany w kolejce transmisji.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_DEAD_LETTER_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

DefClusterXmitQueueTyp (MQLONG)

Atrybut DefClusterXmitQueueType określa, która kolejka transmisji jest domyślnie wybierana przez kanały nadawcze klastra, z których mają być wysyłane komunikaty do kanałów odbiorczych klastra.

Wartościami **DefClusterXmitQueueType** są MQCLXQ_SCTQ albo MQCLXQ_CHANNEL.

MQCLXQ_SCTQ

Wszystkie kanały nadawcze klastra wysyłają komunikaty z produktu SYSTEM.CLUSTER.TRANSMIT.QUEUE. Identyfikator correlID komunikatów umieszczonych w kolejce transmisji wskazuje, do którego kanału nadawczego klastra ma zostać przekazany komunikat.

Parametr SNDJ jest ustawiany podczas definiowania menedżera kolejek.

MQCLXQ_CHANNEL

Każdy kanał nadawczy klastra wysyła komunikaty z innej kolejki transmisji. Każda kolejka transmisji jest tworzona jako trwała kolejka dynamiczna z kolejki modelowej SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Jeśli atrybut menedżera kolejek DefClusterXmitQueueTypejest ustawiony na CHANNEL, Konfiguracja domyślna została zmieniona w taki sposób, że kanały nadawcze klastra zostały powiązane z poszczególnymi kolejkami transmisji klastra. Kolejki transmisji to trwałe kolejki dynamiczne utworzone na podstawie kolejki modelowej SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE. Każda kolejka transmisji jest powiązana z jednym kanałem nadawczym klastra. Ponieważ jeden kanał nadawczy klastra obsługuje kolejkę transmisji klastra, kolejka transmisji zawiera komunikaty dla tylko jednego menedżera kolejek w jednym klastrze. Istnieje możliwość skonfigurowania klastrów w taki sposób, aby każdy menedżer kolejek w klastrze zawierał tylko jedną kolejkę klastra. W takim przypadku ruch komunikatów z menedżera kolejek do każdej kolejki klastra jest przekazywany niezależnie z komunikatów do kolejki.

Aby wysłać zapytanie o wartość, należy wywołać funkcję MQINQlub wysłać komendę Inquire Queue Manager (MQCMD_INQUIRE_Q_MGR) PCF, ustawiając selektor MQIA_DEF_CLUSTER_XMIT_Q_TYPE . Aby zmienić tę wartość, należy wysłać komendę PCF Change Queue Manager (MQCMD_CHANGE_Q_MGR), ustawiając selektor MQIA_DEF_CLUSTER_XMIT_Q_TYPE .

Odsyłacze pokrewne

Zmiana menedżera kolejek

Sprawdź menedżera kolejek

“MQINQ-zapytanie o obiekt-atrybuty” na stronie 727

Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierający atrybuty obiektu.

DefXmitQName (MQCHAR48)

Jest to nazwa kolejki transmisji, która jest używana do transmisji komunikatów do zdalnych menedżerów kolejek, jeśli nie ma innego wskazania, której kolejki transmisji użyć.

Jeśli nie ma domyślnej kolejki transmisji, nazwa jest całkowicie pusta. Początkowa wartość tego atrybutu jest pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_DEF_XMIT_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

DistLists (MQLONG)

Wskazuje, czy menedżer kolejek lokalnych obsługuje listy dystrybucyjne w wywołaniach MQPUT i MQPUT1 . Jest to jedna z następujących wartości:

MQDL_SUPPORTED (obsługiwane)

Obsługiwane listy dystrybucyjne.

NIEOBSŁUGIWANA MQDL_NOT_SUPPORTED

Listy dystrybucyjne nie są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DIST_LISTS z wywołaniem MQINQ.

Grupa DNS (MQCHAR18)

Ten parametr nie jest już używany. Więcej informacji na ten temat zawiera sekcja Co zmieniono w sekcji IBM MQ 8.0.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_DNS_GROUP z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_DNS_GROUP_NAME_LENGTH.

DNSWLM (MQLONG),

Ten parametr nie jest już używany. Więcej informacji na ten temat zawiera sekcja Co zmieniono w sekcji IBM MQ 8.0.

Jest to jedna z następujących wartości:

MQDNSWLM_TAK
Ta wartość może być widoczna w menedżerze kolejek zmigrowanym z wcześniejszej wersji. Wartość jest ignorowana.

MQDNSWLM_NO
Jest to jedyna wartość obsługiwana przez menedżer kolejek.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DNS_WLM z wywołaniem MQINQ.

ExpiryInterval (MQLONG)

Wskazuje częstotliwość, z jaką menedżer kolejek skanuje kolejki w poszukiwaniu komunikatów, które utraciły ważność. Jest to przedział czasu w sekundach z zakresu od 1 do 99 999 999 lub następująca wartość specjalna:

MQEXPI_OFF
Menedżer kolejek nie skanuje kolejek w poszukiwaniu komunikatów, które utraciły ważność.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_EXPIRY_INTERVAL z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

IGQPutAuthority (MQLONG),

Ten atrybut ma zastosowanie tylko wtedy, gdy menedżer kolejek lokalnych jest elementem grupy współużytkowania kolejek. Wskazuje typ sprawdzania uprawnień, które jest wykonywane, gdy lokalny agent kolejkowania wewnątrz grupy (agent IGQ) usuwa komunikat ze współużytkowanej kolejki transmisji i umieszcza komunikat w kolejce lokalnej. Jest to jedna z następujących wartości:

MQIGQPA_DEFAULT
Identyfikator użytkownika sprawdzany pod kątem autoryzacji jest wartością pola *UserIdentifier* w *oddzielnym* deskryptorze MQMD powiązanym z komunikatem, gdy komunikat znajduje się we współużytkowanej kolejce transmisji. Jest to identyfikator użytkownika programu, który umieścił komunikat we współużytkowanej kolejce transmisji i jest zwykle taki sam, jak identyfikator użytkownika, pod którym działa zdalny menedżer kolejek.

Jeśli profil RESLEVEL wskazuje, że ma zostać sprawdzony więcej niż jeden identyfikator użytkownika, sprawdzany jest również identyfikator użytkownika lokalnego agenta IGQ (*IGQUserId*).

MQIGQPA_CONTEXT
Identyfikator użytkownika sprawdzany pod kątem autoryzacji jest wartością pola *UserIdentifier* w *oddzielnym* deskryptorze MQMD powiązanym z komunikatem, gdy komunikat znajduje się we współużytkowanej kolejce transmisji. Jest to identyfikator użytkownika programu, który umieścił komunikat we współużytkowanej kolejce transmisji i jest zwykle taki sam, jak identyfikator użytkownika, pod którym działa zdalny menedżer kolejek.

Jeśli profil RESLEVEL wskazuje, że ma zostać sprawdzony więcej niż jeden identyfikator użytkownika, sprawdzany jest również identyfikator użytkownika lokalnego agenta IGQ (*IGQUserId*) i wartość pola *UserIdentifier* w *wbudowanej* strukturze MQMD. Ostatni identyfikator użytkownika jest zwykle identyfikatorem użytkownika aplikacji, z której pochodzi komunikat.

MQIGQPA_ONLY_IGQ
Identyfikator użytkownika sprawdzany pod kątem autoryzacji jest identyfikatorem użytkownika lokalnego agenta IGQ (*IGQUserId*).

Jeśli profil RESLEVEL wskazuje, że ma zostać sprawdzony więcej niż jeden identyfikator użytkownika, ten identyfikator użytkownika jest używany dla wszystkich sprawdzeń.

MQIGQPA_ALTERNATE_LUB_IGQ
Identyfikator użytkownika sprawdzany pod kątem autoryzacji jest identyfikatorem użytkownika lokalnego agenta IGQ (*IGQUserId*).

Jeśli profil RESLEVEL wskazuje, że ma zostać sprawdzony więcej niż jeden identyfikator użytkownika, sprawdzana jest również wartość pola *UserIdentifier* w *wbudowanej* strukturze MQMD. Ten identyfikator użytkownika jest zwykle identyfikatorem użytkownika aplikacji, z której pochodzi komunikat.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_IGQ_PUT_AUTHORITY z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

IGQUserId (MQLONG)

Ten atrybut ma zastosowanie tylko wtedy, gdy menedżer kolejek lokalnych jest elementem grupy współużytkowania kolejek. Określa on identyfikator użytkownika, który jest powiązany z lokalnym agentem kolejkowania wewnątrz grupy (agent IGQ). Ten identyfikator jest jednym z identyfikatorów użytkowników, które można sprawdzić pod kątem autoryzacji, gdy agent IGQ umieszcza komunikaty w kolejkach lokalnych. Rzeczywiste sprawdzone identyfikatory użytkowników zależą od ustawienia atrybutu **IGQPutAuthority** i od zewnętrznych opcji zabezpieczeń.

Jeśli pole *IGQUserId* jest puste, z agentem IGQ nie jest powiązany żaden identyfikator użytkownika i nie jest wykonywane odpowiednie sprawdzenie autoryzacji (chociaż inne identyfikatory użytkowników mogą być nadal sprawdzane pod kątem autoryzacji).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_IGQ_USER_ID z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_USER_ID_LENGTH.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

InhibitEvent (MQLONG)

Ta opcja określa, czy mają być generowane zdarzenia blokowania (blokowania pobierania i blokowania umieszczania). Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INHIBIT_EVENT z wywołaniem MQINQ.

W systemie z/OSnie można użyć wywołania MQINQ do określenia wartości tego atrybutu.

IntraGroupqueuing (MQLONG)

Ten atrybut ma zastosowanie tylko wtedy, gdy menedżer kolejek lokalnych jest elementem grupy współużytkowania kolejek. Wskazuje, czy kolejkowanie wewnątrz grupy jest włączone dla grupy współużytkowania kolejek. Jest to jedna z następujących wartości:

MQIGQ_WYŁĄCZONE
Wszystkie komunikaty przeznaczone dla innych menedżerów kolejek w grupie współużytkowania kolejek są przesyłane przy użyciu konwencjonalnych kanałów.

MQIGQ_ENABLED
Komunikaty przeznaczone dla innych menedżerów kolejek w grupie współużytkowania kolejek są przesyłane przy użyciu współużytkowanej kolejki transmisji, jeśli spełniony jest następujący warunek:

- Długość danych komunikatu i nagłówka transmisji nie przekracza 63 kB (64 512 bajtów).

  Dla nagłówka transmisji zaleca się przydzielenie większej ilości miejsca niż wielkość MQXQH. W tym celu udostępniono stałą wartość MQ_MSG_HEADER_LENGTH.

Jeśli ten warunek nie jest spełniony, komunikat jest przesyłany za pomocą kanałów konwencjonalnych.

Uwaga: Jeśli kolejkowanie wewnątrz grupy jest włączone, kolejność komunikatów przesyłanych za pomocą współużytkowanej kolejki transmisji nie jest zachowywana względem kolejności komunikatów przesyłanych za pomocą kanałów konwencjonalnych.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INTRA_GROUP_queuing z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

IPAddressVersion (MQLONG)

Określa, która wersja adresu IP (IPv4 lub IPv6) jest używana.

Ten atrybut ma zastosowanie tylko w systemach, w których działają zarówno systemy IPv4 , jak i IPv6 , i ma wpływ na kanały zdefiniowane jako posiadające *TransportType* MQXPY_TCP, jeśli spełniony jest jeden z następujących warunków:

- Kanał *ConnectionName* jest nazwą hosta, która jest tłumaczona zarówno na adres IPv4 , jak i IPv6 , a jego parametr **LocalAddress** nie jest określony.
- Nazwy *ConnectionName* i *LocalAddress* kanału są zarówno nazwami hostów, które są tłumaczone na adresy IPv4 , jak i IPv6 .

Możliwe wartości:

MQIPADDR_IPV4
Używana jest wartość IPv4 .

MQIPADDR_IPV6
Używana jest wartość IPv6 .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_IP_ADDRESS_VERSION z wywołaniem MQINQ.

ListenerTimer (MQLONG)

Jest to przedział czasu (w sekundach) między kolejnymi próbami zrestartowania procesu nasłuchiwania przez program IBM MQ , jeśli wystąpiła awaria APPC lub TCP/IP. Wartość musi być z zakresu od 5 do 9999, a wartością domyślną jest 60.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LISTENER_TIMER z wywołaniem MQINQ.

LocalEvent (MQLONG)

Określa, czy generowane są lokalne zdarzenia błędów. Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LOCAL_EVENT z wywołaniem MQINQ.

W systemie z/OSnie można użyć wywołania MQINQ do określenia wartości tego atrybutu.

LoggerEvent (MQLONG)

Określa, czy generowane są zdarzenia dziennika odtwarzania. Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LOGGER_EVENT z wywołaniem MQINQ.

Multi Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

LUGroupName (MQCHAR8)

Jest to ogólna nazwa jednostki logicznej dla programu nasłuchującego LU 6.2 , który obsługuje transmisje przychodzące dla grupy współużytkowania kolejek. Jeśli ta nazwa pozostanie pusta, nie będzie można używać tego programu nasłuchującego.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_LU_GROUP_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_LU_NAME_LENGTH.

Nazwa jednostki logicznej (MQCHAR8)

Jest to nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji jednostki logicznej 6.2 . Ustaw tę wartość na tę samą jednostkę logiczną, której program nasłuchujący używa do transmisji przychodzących. Jeśli ta nazwa pozostanie pusta, zostanie użyta domyślna jednostka logiczna APPC/MVS; jest to zmienna, dlatego w przypadku korzystania z LU6.2należy zawsze ustawiać wartość LUName.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_LU_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_LU_NAME_LENGTH.

LU62ARMSuffix (MQCHAR2)

Jest to przyrostek SYS1.PARMLIB podzbiór APPCPMxx, który nominuje LUADD dla tego inicjatora kanału. Komenda z/OS SET APPC=xx jest uruchamiana, gdy menedżer ARM restartuje inicjator kanału. Jeśli ta nazwa pozostanie pusta, nie zostanie wydana komenda SET APPC=xx.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_LU62_ARM_SUFFIX z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_ARM_SUFFIX_LENGTH.

LU62Channels (MQLONG)

Jest to maksymalna liczba kanałów, które mogą być kanałami bieżącymi, lub klientów, które mogą być połączone, korzystających z protokołu transmisji LU 6.2 .

Wartość musi należeć do zakresu od 0 do 9999, a wartością domyślną jest 200. Jeśli zostanie ustawiona wartość zero, protokół transmisji jednostki logicznej 6.2 nie będzie używany.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_LU62_CHANNELS z wywołaniem MQINQ.

MaxActivekanałów (MQLONG)

Ten atrybut przedstawia maksymalną liczbę kanałów, które mogą być *aktywne* w dowolnym momencie.

Wartością domyślną jest wartość podana dla atrybutu `MaxChannels`.

W przypadku z/OSwartość musi być z zakresu od 1 do 9 999.

W przypadku wszystkich innych platform wartością domyślną jest 999 999 999, co oznacza, że liczba aktywnych kanałów jest nieograniczona, lub można ją ustawić na rzeczywistą liczbę, aby narzucić limit.

MQ Appliance Nie należy zmieniać wartości **MaxActiveChannels** w systemie IBM MQ Appliance. Aby ograniczyć maksymalną liczbę kanałów klienta, należy użyć atrybutów MAXINST i MAXINSTC dla każdego kanału SVRCONN w celu zdefiniowania ograniczeń dla każdego kanału SVRCONN, patrz sekcja Konfiguracja menedżera kolejek w urządzeniu IBM MQ Appliance w dokumentacji produktu IBM MQ Appliance .

Parametr **MaxActiveChannels** jest atrybutem menedżera kolejek tylko w systemie z/OS . Na innych platformach **MaxActiveChannels** jest atrybutem w pliku `qm.ini` . Informacje na temat ustawiania atrybutu **MaxActiveChannels** na innych platformach zawiera sekcja Sekcje pliku konfiguracyjnego dotyczące kolejkowania rozproszonego .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACTIVE_CHANNELS z wywołaniem **MQINQ** .

Pojęcia pokrewne

Stany kanału

MaxChannels (MQLONG)

Ten atrybut przedstawia maksymalną liczbę kanałów, które mogą być *bieżące* (w tym kanały połączenia z serwerem z połączonymi klientami).

W przypadku z/OSwartość musi należeć do zakresu od 1 do 9 999, a wartością domyślną jest 200.

MQ Appliance W przypadku systemu IBM MQ Appliancewartością domyślną jest 999 999 999 i nie należy jej zmieniać. Aby ograniczyć maksymalną liczbę kanałów klienta, należy użyć atrybutów MAXINST i MAXINSTC dla każdego kanału SVRCONN w celu zdefiniowania ograniczeń dla każdego kanału SVRCONN, patrz sekcja Konfiguracja menedżera kolejek w urządzeniu IBM MQ Appliance w dokumentacji produktu IBM MQ Appliance .

System, który jest zajęty obsługą połączeń z sieci, może wymagać większej liczby niż ustawienie domyślne. Określ wartość, która jest poprawna dla danego środowiska, najlepiej obserwując zachowanie systemu podczas testowania.

W przypadku wszystkich innych platform wartością domyślną jest 100. Można ustawić inną wartość parametru **MaxChannels** , aby ograniczyć maksymalną liczbę bieżących kanałów, jeśli jest to wymagane.

Parametr **MaxChannels** jest atrybutem menedżera kolejek tylko w systemie z/OS . Na innych platformach **MaxChannels** jest atrybutem w pliku `qm.ini` . Informacje na temat ustawiania atrybutu **MaxChannels** na innych platformach zawiera sekcja Sekcje pliku konfiguracyjnego dotyczące kolejkowania rozproszonego .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_CHANNELS z wywołaniem **MQINQ** .

Pojęcia pokrewne

Stany kanału

MaxHandles (MQLONG)

Jest to maksymalna liczba otwartych uchwytów, które mogą być używane jednocześnie przez jedno zadanie. Każde pomyślne wywołanie MQOPEN dla pojedynczej kolejki (lub dla obiektu, który nie jest kolejką) używa jednego uchwytu. Ten uchwyt staje się dostępny do ponownego wykorzystania po zamknięciu obiektu. Jeśli jednak lista dystrybucyjna jest otwarta, każda kolejka na liście dystrybucyjnej ma przydzielony osobny uchwyt, dzięki czemu wywołanie MQOPEN używa tylu uchwytów, ile jest kolejek na liście dystrybucyjnej. Należy to wziąć pod uwagę przy podejmowaniu decyzji o odpowiedniej wartości parametru *MaxHandles*.

Wywołanie MQPUT1 wykonuje wywołanie MQOPEN w ramach przetwarzania. W wyniku tego funkcja MQPUT1 używa tylu uchwytów, ile może użyć komenda MQOPEN, ale uchwyty są używane tylko przez czas trwania wywołania MQPUT1 .

W systemie z/OS *zadanie* oznacza zadanie CICS , zadanie MVS lub region zależny IMS .

Wartość należy do zakresu od 1 do 999 999 999. Wartość domyślna jest określana przez środowisko:

- W systemie z/OSwartością domyślną jest 100.
- We wszystkich innych środowiskach wartością domyślną jest 256.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_UCHWYTY z wywołaniem MQINQ.

MaxMsg(MQLONG)

Jest to długość najdłuższego komunikatu *fizycznego* , który może obsłużyć menedżer kolejek. Jednak ze względu na to, że atrybut menedżera kolejek **MaxMsgLength** można ustawić niezależnie od atrybutu kolejki **MaxMsgLength** , najdłuższy komunikat fizyczny, który można umieścić w kolejce, jest mniejszą z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, aplikacja może umieścić komunikat logiczny, który jest dłuższy niż mniejsza z dwóch atrybutów **MaxMsgLength** , ale tylko wtedy, gdy w deskryptorze MQMD określono opcję MQMF_SEGMENTATION_ALLOWED. Jeśli ta opcja jest określona, górny limit długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zazwyczaj ograniczenia zasobów narzucane przez system operacyjny lub środowisko, w którym działa aplikacja, powodują ograniczenie dolnego limitu.

Dolny limit dla atrybutu **MaxMsgLength** wynosi 32 kB (32 768 bajtów). Górny limit to 100 MB (104 857 600 bajtów).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_MSG_LENGTH z wywołaniem MQINQ.

MaxPriority (MQLONG)

Jest to maksymalny priorytet komunikatu obsługiwany przez menedżer kolejek. Priorytety należą do zakresu od zera (najniższy) do *MaxPriority* (najwyższy).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_PRIORITY z wywołaniem MQINQ.

MaxPropertiesDługość (MQLONG)

Służy do sterowania wielkością właściwości, które mogą przepływać z komunikatem. Obejmuje to zarówno nazwę właściwości w bajtach, jak i wielkość wartości właściwości w bajtach.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_PROPERTIES_LENGTH z wywołaniem MQINQ.

Multi *MaxQFile(MQLONG)*

Maksymalna wielkość (w megabajtach), do której może zostać powiększony plik kolejki.

Tabela 560. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Plik kolejki może przekroczyć maksymalną wielkość, jeśli jest on skonfigurowany na wartość mniejszą niż bieżąca wielkość pliku kolejki. W takim przypadku plik kolejki nie akceptuje już nowych komunikatów, ale zezwala na wykorzystanie istniejących komunikatów. Jeśli wielkość pliku kolejki spadła poniżej skonfigurowanej wartości, nowe komunikaty mogą być umieszczane w kolejce.

Uwaga: Ten rysunek może różnić się od wartości atrybutu skonfigurowanego w kolejce, ponieważ wewnętrznie menedżer kolejek może potrzebować większego bloku, aby osiągnąć wybraną wielkość. Więcej informacji na temat zmiany wielkości plików kolejki oraz wielkości bloku i granulacji zawiera sekcja Modyfikowanie plików kolejki produktu IBM MQ .

Jeśli granulacja wymaga zmiany z powodu zwiększenia tego atrybutu, w dziennikach AMQERR zapisywany jest komunikat ostrzegawczy AMQ7493W `Granularity changed` (Zmieniona granulacja). Oznacza to, że należy zaplanować opróżnianie kolejki, aby produkt IBM MQ mógł przyjąć nową granulację.

Maksymalna wartość tego atrybutu to 267,386,880 MB, a wartość domyślna i wartość zmigrowana to 2,088,960 MB. Jest to bieżące maksimum dla kolejki o granulacji równej 512.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_Q_FILE_SIZE z wywołaniem MQINQ.

MaxUncommittedkomunikatów (MQLONG)

Jest to maksymalna liczba niezatwierdzonych komunikatów, które mogą istnieć w jednostce pracy. Liczba niezatwierdzonych komunikatów jest sumą następujących wartości od momentu rozpoczęcia bieżącej jednostki pracy:

- Komunikaty umieszczane przez aplikację z opcją MQPMO_SYNCPOINT
- Komunikaty pobierane przez aplikację z opcją MQGMO_SYNCPOINT
- Komunikaty wyzwalacza i komunikaty raportu COA wygenerowane przez menedżer kolejek dla komunikatów umieszczonych za pomocą opcji MQPMO_SYNCPOINT
- Komunikaty raportu COD generowane przez menedżera kolejek dla komunikatów pobranych z opcją MQGMO_SYNCPOINT

Następujące komunikaty nie są liczone jako niezatwierdzone:

- Komunikaty umieszczone lub pobrane przez aplikację poza jednostką pracy
- Komunikaty wyzwalacza lub COA/COD raportują komunikaty wygenerowane przez menedżera kolejek w wyniku umieszczenia lub pobrania komunikatów poza jednostką pracy
- Komunikaty raportu o utracie ważności wygenerowane przez menedżer kolejek (nawet jeśli wywołanie powodujące komunikat raportu o utracie ważności określały parametr MQGMO_SYNCPOINT)
- Komunikaty zdarzeń wygenerowane przez menedżer kolejek (nawet jeśli wywołanie powodujące komunikat zdarzenia określały MQPMO_SYNCPOINT lub MQGMO_SYNCPOINT)

Uwaga:

1. Komunikaty raportu o wyjątkach są generowane przez agent kanału komunikatów (MCA) lub przez aplikację i są traktowane w taki sam sposób, jak zwykłe komunikaty umieszczane lub pobierane przez aplikację.
2. Jeśli komunikat lub segment jest umieszczany z opcją MQPMO_SYNCPOINT, liczba niezatwierdzonych komunikatów jest zwiększana o jeden niezależnie od liczby komunikatów fizycznych, które są rzeczywiście wynikiem umieszczenia. (Jeśli menedżer kolejek musi podzielić komunikat lub segment, może zostać wyświetlony więcej niż jeden komunikat fizyczny).
3. Gdy lista dystrybucyjna jest umieszczana z opcją MQPMO_SYNCPOINT, liczba niezatwierdzonych komunikatów jest zwiększana o jeden *dla każdego wygenerowanego komunikatu fizycznego*. Liczba ta może być tak mała, jak jedna, lub tak duża, jak liczba miejsc docelowych na liście dystrybucyjnej.

Dolny limit dla tego atrybutu to 1; górny limit to 999 999 999. Wartością domyślną jest 10000.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_UNCOMMITTED_MSGS z wywołaniem MQINQ.

Rozliczanie MQI (MQLONG)

Steruje gromadzeniem informacji rozliczeniowych dla danych MQI.

Jest to jedna z następujących wartości:

MQMON_ON
Zgromadź dane rozliczeniowe interfejsu API.

MQMON_WYŁ
Nie zbieraj danych rozliczeniowych interfejsu API. Jest to wartość domyślna.

Jeśli atrybut ACCTCONO menedżera kolejek zostanie ustawiony na wartość ENABLED, ta wartość może zostać nadpisana dla pojedynczych połączeń przy użyciu pola Opcje w strukturze MQCNO. Zmiany tej wartości obowiązują tylko w przypadku połączeń z menedżerem kolejek, które wystąpiły po zmianie atrybutu.

Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_MQI z wywołaniem MQINQ.

Statystyki MQI (MQLONG)

Steruje to gromadzeniem informacji dotyczących monitorowania statystyk dla menedżera kolejek.

Jest to jedna z następujących wartości:

MQMON_ON
Zgromadź statystyki MQI.

MQMON_WYŁ
Nie zbieraj statystyk MQI. Jest to wartość domyślna.

Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_MQI z wywołaniem MQINQ.

MsgMarkBrowseInterval (MQLONG)

Odstęp czasu (w milisekundach), po upływie którego menedżer kolejek może automatycznie usunąć znacznik z komunikatów przeglądania.

Jest to odstęp czasu (w milisekundach), po upływie którego menedżer kolejek może automatycznie usunąć znacznik z komunikatów przeglądania.

Ten atrybut opisuje przedział czasu, przez który komunikaty oznaczone jako przejrzane przez wywołanie MQGET z użyciem opcji pobierania komunikatu MQGMO_MARK_BROWSE_CO_OP powinny pozostać oznaczone jako przejrzane.

Menedżer kolejek może automatycznie usunąć zaznaczenie przejrzanych komunikatów, które zostały oznaczone jako przejrzane dla współpracującego zestawu uchwytów, jeśli zostały oznaczone na więcej niż ten przybliżony przedział czasu.

Nie ma to wpływu na stan żadnego komunikatu oznaczonego jako przeglądanie, który został uzyskany przez wywołanie MQGET przy użyciu opcji pobierania komunikatu MQGMO_MARK_BROWSE_HANDLE.

Wartością maksymalną jest 999 999 999, a wartością domyślną jest 5000. Wartość specjalna -1 dla parametru *MsgMarkBrowseInterval* oznacza nieograniczony przedział czasu.

Ostrzeżenie: Ta wartość nie powinna być mniejsza niż wartość domyślna 5000.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MSG_MARK_BROWSE_INTERVAL z wywołaniem MQINQ.

OutboundPortMaks. (MQLONG)

Jest to najwyższy numer portu w zakresie zdefiniowanym przez wartości minimalne OutboundPorti OutboundPortdla numerów portów, które mają być używane do wiązania kanałów wychodzących.

Wartość jest liczbą całkowitą z zakresu od 0 do 65535 i musi być równa lub większa od wartości minimalnej OutboundPort. Wartością domyślną jest 0.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OUTBOUND_PORT_MAX z wywołaniem MQINQ.

OutboundPortMin. (MQLONG)

Jest to najniższy numer portu w zakresie zdefiniowanym przez wartości minimalne OutboundPorti OutboundPortdla numerów portów, które mają być używane do wiązania kanałów wychodzących.

Wartość jest liczbą całkowitą z zakresu od 0 do 65535 i musi być równa lub mniejsza od wartości maksymalnej OutboundPort. Wartością domyślną jest 0.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OUTBOUND_PORT_MIN z wywołaniem MQINQ.

PerformanceEvent (MQLONG) (zdarzenie wykonania)

Ta opcja określa, czy generowane są zdarzenia związane z wydajnością. Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PERFORMANCE_EVENT z wywołaniem MQINQ.

Platforma (MQLONG)

Wskazuje system operacyjny, w którym działa menedżer kolejek:

MQPL_AIX
AIX (taka sama jak MQPL_UNIX).

MQPL_APPLIANCE
IBM MQ Appliance

MQPL_MVS
z/OS (taka sama jak MQPL_ZOS).

MQPL_OS390
z/OS (taka sama jak MQPL_ZOS).

MQPL_OS400
IBM i.

MQPL_UNIX
UNIX.

MQPL_WINDOWS_NT
Windows stowarzyszonych.

MQPL_ZOS
z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PLATFORM z wywołaniem MQINQ.

PubSubNPInputMsg (MQLONG)

Określa, czy odrzucić, czy zachować niedostarczony komunikat wejściowy.

Jest to jedna z następujących wartości:

MQUNDELIVERED_DISCARD,
Nietrwałe komunikaty wejścia mogą być usuwane, jeśli nie można ich przetworzyć.

Jest to wartość domyślna.

MQUNDELIVERED_KEEP
Nietrwałe komunikaty wejścia nie będą usuwane, jeśli nie można ich przetworzyć. W takiej sytuacji umieszczony w kolejce interfejs publikowania/subskrypcji będzie ponawiał proces w odpowiednich odstępach czasu i nie będzie kontynuował przetwarzania kolejnych komunikatów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_NP_MSG z wywołaniem MQINQ.

PubSubOdpowiedź NPResponse (MQLONG)
Steruje zachowaniem niedostarczonych komunikatów odpowiedzi.

Jest to jedna z następujących wartości:

MQUNDELIVERED_NORMAL
Nietrwałe odpowiedzi, których nie można umieścić w kolejce odpowiedzi, są umieszczane w kolejce niedostarczonych komunikatów, jeśli nie można ich umieścić w kolejce DLQ, są usuwane.

MQUNDELIVERED_SAFE
Nietrwałe odpowiedzi, których nie można umieścić w kolejce odpowiedzi, są umieszczane w kolejce niedostarczonych komunikatów. Jeśli nie można ustawić odpowiedzi i nie można jej umieścić w kolejce DLQ, to umieszczony w kolejce interfejs publikowania/subskrypcji wycofa bieżącą operację, a następnie ponowi próbę w odpowiednich odstępach czasu i nie będzie kontynuował przetwarzania kolejnych komunikatów.

MQUNDELIVERED_DISCARD,
Nietrwałe odpowiedzi nie są umieszczane w kolejce odpowiedzi i są odrzucane.

Jest to wartość domyślna dla nowych menedżerów kolejek.

MQUNDELIVERED_KEEP
Nietrwałe odpowiedzi nie są umieszczane w kolejce niedostarczonych komunikatów ani odrzucane. Zamiast tego umieszczony w kolejce interfejs publikowania/subskrypcji wycofa bieżącą operację, a następnie ponowi ją w odpowiednich odstępach czasu.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_NP_RESP z wywołaniem MQINQ.

Wartość domyślna dla migrowanych menedżerów kolejek.
Jeśli menedżer kolejek został zmigrowany z wersji IBM MQ V6.0, początkowa wartość tego atrybutu zależy od wartości atrybutu *DiscardNonPersistentResponse* i *DLQNonPersistentResponse* przed migracją, co przedstawiono w poniższej tabeli.

		Odpowiedź DLQNonPersistent		
		Tak	**Nie**	**Nie ustawiono**
DiscardNonPersistentResponse	**Tak**	MQUNDELIVERED_NORMAL	MQUNDELIVERED_DISCARD,	MQUNDELIVERED_NORMAL
	Nie	MQUNDELIVERED_SAFE	MQUNDELIVERED_KEEP	MQUNDELIVERED_SAFE
	Nie ustawiono	Jeśli parametr SyncPointma wartość Persistent = No, MQUNDELIVERED_SAFE, W przeciwnym razie MQUNDELIVERED_NORMAL	Jeśli parametr SyncPointma wartość Persistent = No, MQUNDELIVERED_KEEP else MQUNDELIVERED_DISCARD	Jeśli parametr SyncPointma wartość Persistent = No, MQUNDELIVERED_SAFE, W przeciwnym razie MQUNDELIVERED_NORMAL

PubSubMaxMsgRetryCount (MQLONG)
Liczba prób podczas przetwarzania komunikatu komendy zakończonej niepowodzeniem w punkcie synchronizacji.

Jest to jedna z następujących wartości:

0-999 999 999
Wartość domyślna to 5.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_MAXMSG_RETRY_COUNT z wywołaniem MQINQ.

PubSubSyncPoint (MQLONG)

Określa, czy tylko komunikaty trwałe, czy wszystkie komunikaty są przetwarzane w punkcie synchronizacji.

Jest to jedna z następujących wartości:

MQSYNCPOINT_IFPER
Powoduje to, że umieszczony w kolejce interfejs publikowania/subskrypcji odbiera nietrwałe komunikaty poza punktem synchronizacji. Jeśli demon odbierze publikację spoza punktu synchronizacji, przekazuje ją do znanych subskrybentów znajdujących się poza punktem synchronizacji.

Jest to wartość domyślna.

MQSYNCPOINT_YES
Powoduje to, że umieszczony w kolejce interfejs publikowania/subskrypcji odbiera wszystkie komunikaty w punkcie synchronizacji.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_SYNC_PT z wywołaniem MQINQ.

Tryb PubSub(MQLONG)

Określa, czy mechanizm publikowania/subskrypcji i umieszczony w kolejce interfejs publikowania/subskrypcji są uruchomione, co umożliwia aplikacjom publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania.

Jest to jedna z następujących wartości:

MQPSM_COMPAT
Mechanizm publikowania/subskrybowania działa. W związku z tym możliwe jest publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego. Umieszczony w kolejce interfejs publikowania/subskrypcji nie jest uruchomiony, dlatego nie są wykonywane żadne działania na komunikatach umieszczanych w kolejkach monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania. To ustawienie jest używane w celu zachowania zgodności z produktem WebSphere Message Broker V6 lub wcześniejszej używającym tego menedżera kolejek, ponieważ musi odczytywać te same kolejki, z których normalnie odczytywany jest umieszczony w kolejce interfejs publikowania/subskrypcji.

MQPSM_WYŁĄCZONE
Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania nie działają. Dlatego nie jest możliwe publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego. Żadne komunikaty publikowania/subskrypcji, które są umieszczane w kolejkach monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania, nie są uwzględniane.

MQPSM_ENABLED
Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania działają. W związku z tym możliwe jest publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania. Jest to początkowa wartość domyślna menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_PUBSUB_MODE z wywołaniem MQINQ.

QMgrDesc (MQCHAR64)

To pole służy do komentowania menedżera kolejek. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony

odstępami. W instalacji DBCS pole to może zawierać znaki DBCS (maksymalna długość pola wynosi 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

- W systemie z/OSwartością domyślną jest nazwa produktu i numer wersji.
- We wszystkich innych środowiskach wartością domyślną jest wartość pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_MGR_DESC z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_MGR_DESC_LENGTH.

QMgrIdentifier (MQCHAR48)

Jest to wygenerowana wewnętrznie nazwa unikalna dla menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_MGR_IDENTIFIER z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_MGR_IDENTIFIER_LENGTH.

Ten atrybut jest obsługiwany w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

i klientów IBM MQ połączonych z tymi systemami.

QMgrName (MQCHAR48)

Jest to nazwa lokalnego menedżera kolejek, czyli nazwa menedżera kolejek, z którym połączona jest aplikacja.

Pierwsze 12 znaków nazwy jest używanych do konstruowania unikalnego identyfikatora komunikatu (patrz MQMD- MsgId pole). Menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się pierwszymi 12 znakami, aby identyfikatory komunikatów były unikalne w sieci menedżerów kolejek.

W systemie z/OSnazwa jest taka sama jak nazwa podsystemu, która jest ograniczona do 4 niepustych znaków.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_MGR_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

Nazwa QSGName (MQCHAR4)

Jest to nazwa grupy współużytkowania kolejek, do której należy menedżer kolejek lokalnych. Jeśli lokalny menedżer kolejek nie należy do grupy współużytkowania kolejek, nazwa jest pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_QSG_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_QSG_NAME_LENGTH.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

QueueAccounting (MQLONG),

Steruje to gromadzeniem informacji rozliczeniowych dla kolejek.

Jest to jedna z następujących wartości:

MQMON_BRAK
Nie zbieraj danych rozliczeniowych dla kolejek, niezależnie od ustawienia atrybutu rozliczania kolejki ACCTQ. Jest to wartość domyślna.

MQMON_WYŁ
Nie zbieraj danych rozliczeniowych dla kolejek, które określają QMGR w atrybucie kolejki ACCTQ.

MQMON_ON
Zgromadź dane rozliczeniowe dla kolejek, które określają QMGR w atrybucie kolejki ACCTQ.

Zmiany tej wartości obowiązują tylko w przypadku połączeń z menedżerem kolejek, które wystąpiły po zmianie atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_Q z wywołaniem MQINQ.

QueueMonitoring (MQLONG)

Określa domyślne ustawienie monitorowania kolejek w trybie z połączeniem.

Jeśli atrybut kolejki **QueueMonitoring** jest ustawiony na wartość MQMON_Q_MGR, ten atrybut określa wartość, która jest przyjmowana przez kanał. Możliwe wartości:

MQMON_WYŁ
Gromadzenie danych monitorowania w trybie z połączeniem jest wyłączone. Jest to początkowa wartość domyślna menedżera kolejek.

MQMON_BRAK
Kolekcjonowanie danych monitorowania bezpośredniego jest wyłączone dla kolejek niezależnie od ustawienia ich atrybutu **QueueMonitoring** .

MQMON_NISKI
Gromadzenie danych monitorowania bezpośredniego jest włączone, przy niskim współczynniku gromadzenia danych.

MQMON_MEDIUM
Gromadzenie danych monitorowania w trybie z połączeniem jest włączone, przy umiarkowanym współczynniku gromadzenia danych.

MQMON_HIGH
Gromadzenie danych monitorowania bezpośredniego jest włączone, przy wysokim współczynniku gromadzenia danych.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_Q z wywołaniem MQINQ.

QueueStatistics (MQLONG)

Steruje to gromadzeniem danych statystycznych dla kolejek.

Jest to jedna z następujących wartości:

MQMON_BRAK
Nie kolekcjonuj statystyk kolejek bez względu na ustawienie atrybutu kolejki **QueueStatistics** . Jest to wartość domyślna.

MQMON_WYŁ
Nie należy gromadzić danych statystycznych dla kolejek, które określają menedżer kolejek w atrybucie kolejki **QueueStatistics** .

MQMON_ON
Zgromadź dane statystyczne dla kolejek, które określają menedżera kolejek w atrybucie kolejki **QueueStatistics** .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_Q z wywołaniem MQINQ.

ReceiveTimeout (MQLONG)

Określa, jak długo kanał TCP/IP oczekuje na odbiór danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego. Dotyczy to tylko kanałów komunikatów, a nie kanałów MQI.

Dokładne znaczenie parametru ReceiveTimeout jest zmieniane przez wartość określoną przez typ ReceiveTimeout. ReceiveTimeoutTyp można ustawić na jedną z następujących wartości:

- MQRCVTIME_EQUAL-ta wartość określa liczbę sekund oczekiwania kanału. Podaj wartość z zakresu od 0 do 999999.
- MQRCVTIME_ADD-jest to liczba sekund, która ma zostać dodana do negocjowanej wartości HBINT, i określa czas oczekiwania kanału. Należy podać wartość z zakresu od 1 do 999999.
- MQRCVTIME_MULTIPLY-ta wartość jest mnożnikiem, który ma zostać zastosowany do wynegocjowanego HBINT. Podaj wartość 0 lub wartość z zakresu od 2 do 99.

Wartością domyślną jest 0.

Ustaw typ ReceiveTimeout(Limit czasu odbierania) na MQRCVTIME_MULTIPLY lub MQRCVTIME_EQUAL, a wartość ReceiveTimeout na 0, aby zatrzymać przekroczenie przez kanał limitu czasu oczekiwania na odebranie danych od partnera.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RECEIVE_TIMEOUT z wywołaniem MQINQ.

ReceiveTimeoutMin. (MQLONG)

Jest to minimalny czas w sekundach, przez który kanał TCP/IP oczekuje na odbiór danych, w tym pulsów, od partnera, przed powrotem do stanu nieaktywnego.

Dotyczy to tylko kanałów komunikatów, a nie kanałów MQI. Wartość musi być z zakresu od 0 do 999999 (wartość domyślna to 0).

Jeśli typ ReceiveTimeout(Limit czasu odbierania) jest używany do określenia, że czas oczekiwania kanału TCP/IP ma być obliczany względem wynegocjowanej wartości HBINT, a wartość wynikowa jest mniejsza niż wartość tego parametru, to zamiast niej używana jest ta wartość.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RECEIVE_TIMEOUT_MIN z wywołaniem MQINQ.

ReceiveTimeoutTyp (MQLONG)

Jest to kwalifikator stosowany dla parametru ReceiveTimeout w celu zdefiniowania czasu oczekiwania kanału TCP/IP na odebranie danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego. Dotyczy to tylko kanałów komunikatów, a nie kanałów MQI.

Jest to jedna z następujących wartości:

MQRCVTIME_MULTIPLY
ReceiveTimeout : Mnożnik stosowany do negocjowanej wartości HBINT w celu określenia czasu oczekiwania kanału. Jest to wartość domyślna.

MQRCVTIME_ADD
ReceiveTimeout (Limit czasu odbioru) to wartość, w sekundach, dodawana do wynegocjowanej wartości HBINT w celu określenia czasu oczekiwania kanału.

MQRCVTIME_EQUAL
ReceiveTimeout (Limit czasu odbierania) to wartość (w sekundach), przez którą kanał oczekuje.

Aby zatrzymać limit czasu oczekiwania kanału na odebranie danych od partnera, należy ustawić wartość parametru ReceiveTimeoutna MQRCVTIME_MULTIPLY lub MQRCVTIME_EQUAL, a wartość parametru ReceiveTimeout na 0.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RECEIVE_TIMEOUT_TYPE z wywołaniem MQINQ.

RemoteEvent (MQLONG)

Ta opcja określa, czy mają być generowane zdalne zdarzenia błędów. Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_REMOTE_EVENT z wywołaniem MQINQ.

RepositoryName (MQCHAR48)

Jest to nazwa klastra, dla którego ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę dla więcej niż jednego klastra, parametr *RepositoryNamelist* określa nazwę obiektu listy nazw, który identyfikuje klastry, a parametr *RepositoryName* jest pusty. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REPOSITORY_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

RepositoryNamelist (MQCHAR48)

Jest to nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę tylko dla jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć parametru *RepositoryName* , aby określić nazwę klastra, w którym to przypadku wartość *RepositoryNamelist* jest pusta. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REPOSITORY_NAMELIST z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_NAMELIST_NAME_LENGTH.

ScyCase(MQCHAR8)

Określa, czy menedżer kolejek obsługuje nazwy profili zabezpieczeń zapisane literami o różnej wielkości, czy tylko wielkimi literami.

Jest to jedna z następujących wartości:

MQSCYC_UPPER
Nazwy profili zabezpieczeń muszą być zapisane wielkimi literami.

MQSCYC_MIXED
Nazwy profili zabezpieczeń mogą być pisane wielkimi literami lub literami o różnej wielkości.

Zmiany tego atrybutu są uwzględniane po uruchomieniu komendy Odśwież zabezpieczenia z określoną wartością *SecurityType(MQSECTYPE_CLASSES)* .

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SECURITY_CASE z wywołaniem MQINQ.

SharedQMgrNazwa (MQLONG)

Określa, czy program *ObjectQmgrName* powinien być używany, czy traktowany jako lokalny menedżer kolejek w wywołaniu MQOPEN dla kolejki współużytkowanej, gdy program *ObjectQmgrName* jest menedżerem innego menedżera kolejek w grupie współużytkowania kolejek.

Możliwe wartości:

MQSQQM_UŻYJ
Zostanie użyta wartość *ObjectQmgrName* i zostanie otwarta odpowiednia kolejka transmisji.

MQSQQM_IGNORE,
Jeśli kolejka docelowa jest współużytkowana, a *ObjectQmgrName* jest menedżerem kolejek w tej samej grupie współużytkowania kolejek, otwarcie jest wykonywane lokalnie.

Ten atrybut jest poprawny tylko w systemie z/OS.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SHARED_Q_Q_MGR_NAME z wywołaniem MQINQ.

SPLCAP
Wskazuje, czy możliwości zabezpieczeń produktu Advanced Message Security są dostępne dla menedżera kolejek.

OBSŁUGI_MQCAP_SUPPORTED
Jest to wartość domyślna, jeśli komponent AMS jest zainstalowany dla instalacji, w której działa menedżer kolejek.

NIEOBSŁUGIWANA MQCAP_NOT_SUPPORTED

SSLEvent (MQLONG)
Określa, czy są generowane zdarzenia TLS.

Jest to jedna z następujących wartości:

MQEVR_ENABLED,
Wygeneruj zdarzenia TLS w następujący sposób:

MQRC_CHANNEL_SSL_ERROR

MQEVR_DISABLED (wyłączone)
Nie generuj zdarzeń TLS. Jest to wartość domyślna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SSL_EVENT z wywołaniem MQINQ.

Wymagany protokół SSLFIPS(MQLONG)

Uwaga: W systemie AIX, Linux, and Windows IBM MQ zapewnia zgodność ze standardem FIPS 140-2 za pośrednictwem modułu szyfrującego IBM Crypto for C (ICC) . Certyfikat dla tego modułu został przeniesiony do statusu historycznego. Klienci powinni zapoznać się z informacjami w sekcji Certyfikat IBM Crypto for C (ICC) i zapoznać się z poradami NIST. Zastępczy moduł FIPS 140-3 jest obecnie w toku, a jego status można wyświetlić, wyszukując go na liście Moduły NIST CMVP na liście procesów.

Umożliwia to określenie, że mają być używane tylko algorytmy z certyfikatem FIPS, jeśli szyfrowanie jest wykonywane w produkcie IBM MQ, a nie w sprzęcie szyfrującym. Jeśli sprzęt szyfrujący jest skonfigurowany, używane moduły kryptograficzne są modułami udostępnianymi przez produkt sprzętowy. Moduły te mogą, ale nie muszą, mieć certyfikat FIPS na określonym poziomie, w zależności od używanego produktu sprzętowego.

Wartość jest jedną z następujących wartości:

MQSSL_FIPS_NO
Użyj dowolnej CipherSpec obsługiwanej na używanej platformie. Jest to wartość domyślna.

MQSSL_FIPS_YES
W polu CipherSpecs należy używać tylko algorytmów szyfrowania z certyfikatem FIPS, które są dozwolone we wszystkich połączeniach TLS z i do tego menedżera kolejek.

Ten parametr jest poprawny tylko na platformach z/OSi AIX, Linux, and Windows .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SSL_FIPS_REQUIRED z wywołaniem MQINQ.

Zadania pokrewne
Określenie, że w czasie wykonywania na kliencie MQI będą używane tylko CipherSpecs z certyfikatem FIPS.

Odsyłacze pokrewne

Standardy FIPS (Federal Information Processing Standards) dla AIX, Linux, and Windows

Liczba operacji SSLKeyReset(MQLONG)

Określa, kiedy agenty kanału komunikatów TLS (MCA) inicjujące komunikację resetują klucz tajny używany do szyfrowania kanału.

Wartość reprezentuje całkowitą liczbę nieszyfrowanych bajtów, które są wysyłane i odbierane za pomocą kanału przed renegocjacją klucza tajnego. Liczba bajtów obejmuje informacje sterujące wysłane przez agenta MCA.

Wartość jest liczbą z zakresu od 0 do 999 999 999, z wartością domyślną 0. Jeśli zostanie podana liczba operacji resetowania tajnego klucza TLS z zakresu od 1 bajtu do 32 kB, kanały TLS będą używać liczby operacji resetowania tajnego klucza o wielkości 32 kB. Ma to na celu uniknięcie kosztów przetwarzania nadmiernej liczby operacji resetowania klucza, które miałyby miejsce w przypadku małych wartości resetowania tajnego klucza TLS.

Klucz tajny jest renegocjowany, gdy łączna liczba niezaszyfrowanych bajtów wysłanych i odebranych przez agent MCA kanału inicjującego przekroczy określoną wartość. Jeśli puls kanału jest włączony, klucz tajny jest renegocjowany przed wysłaniem lub odebraniem danych zgodnie z pulsem kanału lub gdy łączna liczba niezaszyfrowanych bajtów przekracza określoną wartość, w zależności od tego, która z tych wartości jest pierwsza.

Liczba bajtów wysłanych i odebranych w celu renegocjacji obejmuje informacje sterujące wysłane i odebrane przez MCA kanału i jest resetowana przy każdym wystąpieniu renegocjacji.

Użyj wartości 0, aby wskazać, że klucze tajne nigdy nie są renegocjowane.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SSL_RESET_COUNT z wywołaniem MQINQ.

Zdarzenie StartStop(MQLONG)

Ta opcja określa, czy mają być generowane zdarzenia uruchomienia i zatrzymania. Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_START_STOP_EVENT z wywołaniem MQINQ.

StatisticsInterval (MQLONG)

Określa, jak często (w sekundach) mają być zapisywane dane monitorowania statystyk w kolejce monitorowania.

Wartością jest liczba całkowita z zakresu od 0 do 604800, a wartością domyślną jest 1800 (30 minut).

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_STATISTICS_INTERVAL z wywołaniem MQINQ.

SyncPoint (MQLONG)

Wskazuje, czy lokalny menedżer kolejek obsługuje jednostki pracy i synchronizowanie z wywołaniami MQGET, MQPUT i MQPUT1 .

MQSP_DOSTĘPNE
Dostępne jednostki pracy i synchronizowanie.

MQSP_NIEDOSTĘPNE
Jednostki pracy i synchronizowanie nie są dostępne.

- W systemie z/OS ta wartość nigdy nie jest zwracana.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SYNCPOINT z wywołaniem MQINQ.

Kanały TCP (MQLONG)

Jest to maksymalna liczba kanałów, które mogą być kanałami bieżącymi, lub klientów, które mogą być połączone, korzystających z protokołu transmisji TCP/IP.

Wartość musi należeć do zakresu od 0 do 9999, a wartością domyślną jest 200. Jeśli zostanie podana wartość 0, protokół TCP/IP nie będzie używany.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TCP_CHANNELS z wywołaniem MQINQ.

TCPKeepAlive (MQLONG)

Określa, czy użyć TCP KEEPALIVE do sprawdzenia, czy drugi koniec połączenia jest nadal dostępny. Jeśli drugi koniec połączenia nie jest dostępny, kanał zostanie zamknięty.

Jest to jedna z następujących wartości:

MQTCPKEEP_YES
Użyj parametru TCP KEEPALIVE określonego w zestawie danych konfiguracji profilu TCP. Jeśli zostanie podany atrybut kanału KeepAliveInterwał (KAINT), zostanie użyta wartość, na którą jest on ustawiony.

MQTCPKEEP_NO
Nie należy używać protokołu TCP KEEPALIVE. Jest to wartość domyślna.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TCP_KEEP_ALIVE z wywołaniem MQINQ.

Nazwa TCP (MQCHAR8)

Jest to nazwa jedynego lub preferowanego stosu TCP/IP, który będzie używany, w zależności od wartości TCPStackType. Ten parametr ma zastosowanie tylko w środowiskach CINET z wieloma stosami. Wartością domyślną jest TCPIP.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_TCP_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_TCP_NAME_LENGTH.

TCPStackType (MQLONG),

Określa, czy inicjator kanału może używać tylko stosu TCP/IP określonego w parametrze TCPName, czy też opcjonalnie może być powiązany z dowolnym wybranym stosem TCP/IP. Ten parametr ma zastosowanie tylko w środowiskach CINET z wieloma stosami.

Jest to jedna z następujących wartości:

MQTCPSTACK_SINGLE
Inicjator kanału może używać tylko przestrzeni adresowych TCP/IP wymienionych w polu TCPName. Jest to wartość domyślna.

MQTCPSTACK_MULTIPLE
Inicjator kanału może użyć dowolnej dostępnej dla niego przestrzeni adresowej TCP/IP. Wartością domyślną jest nazwa określona w parametrze TCPName, jeśli dla kanału lub programu nasłuchującego nie określono innej nazwy.

Ten atrybut jest obsługiwany tylko w systemie z/OS .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TCP_STACK_TYPE z wywołaniem MQINQ.

TraceRouteRejestrowanie (MQLONG)

Steruje to rejestrowaniem informacji o trasie śledzenia.

Jest to jedna z następujących wartości:

MQRECORDING_DISABLED,
Dopisywanie do komunikatów trasy śledzenia nie jest dozwolone.

MQRECORDING_Q (kolejka MQ)
Umieść komunikaty trasy śledzenia w stałej kolejce nazwanej.

MQRECORDING_MSG
Umieść komunikaty trasy śledzenia w kolejce określonej za pomocą samego komunikatu. Jest to wartość domyślna

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRACE_ROUTE_RECORDING z wywołaniem MQINQ.

TriggerInterval (MQLONG)

Jest to przedział czasu (w milisekundach) używany do ograniczenia liczby komunikatów wyzwalacza. Ma to zastosowanie tylko wtedy, gdy parametr *TriggerType* ma wartość MQTT_FIRST. W takim przypadku komunikaty wyzwalacza są zwykle generowane tylko wtedy, gdy w kolejce pojawi się odpowiedni komunikat, a kolejka była wcześniej pusta. Jednak w pewnych okolicznościach można wygenerować dodatkowy komunikat wyzwalacza z wyzwalaniem MQTT_FIRST, nawet jeśli kolejka nie była pusta. Te dodatkowe komunikaty wyzwalacza nie są generowane częściej niż co *TriggerInterval* milisekund.

Więcej informacji na temat wyzwalania zawiera sekcja Wyzwalanie kanałów.

Wartość jest nie mniejsza niż 0 i nie większa niż 999 999 999. Wartością domyślną jest 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_INTERVAL z wywołaniem MQINQ.

TriggerInterval (MQLONG)

Jest to przedział czasu (w milisekundach) używany do ograniczenia liczby komunikatów wyzwalacza. Ma to zastosowanie tylko wtedy, gdy parametr *TriggerType* ma wartość MQTT_FIRST. W takim przypadku komunikaty wyzwalacza są zwykle generowane tylko wtedy, gdy w kolejce pojawi się odpowiedni komunikat, a kolejka była wcześniej pusta. Jednak w pewnych okolicznościach można wygenerować dodatkowy komunikat wyzwalacza z wyzwalaniem MQTT_FIRST, nawet jeśli kolejka nie była pusta. Te dodatkowe komunikaty wyzwalacza nie są generowane częściej niż co *TriggerInterval* milisekund.

Więcej informacji na temat wyzwalania zawiera sekcja Wyzwalanie kanałów.

Wartość jest nie mniejsza niż 0 i nie większa niż 999 999 999. Wartością domyślną jest 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_INTERVAL z wywołaniem MQINQ.

Wersja (MQCFST)

Jest to wersja kodu IBM MQ w formacie VVRRMMFF, gdzie:

VV-wersja

RR-zwolnienie

MM-poziom konserwacyjny

FF-poziom poprawek

XrCapability(MQLONG)

Określa, czy komendy MQ Telemetry są obsługiwane przez menedżer kolejek.

Jest to jedna z następujących wartości:

OBSŁUGI_MQCAP_SUPPORTED
Obsługiwane są zainstalowane komponenty MQ Telemetry i komendy produktu Telemetry.

NIEOBSŁUGIWANA MQCAP_NOT_SUPPORTED
Komponent MQ Telemetry nie został zainstalowany.

Ten atrybut jest obsługiwany tylko w systemie Wiele platform.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_XR_CAPABILITY z wywołaniem MQINQ .

Atrybuty kolejek

Istnieje pięć typów definicji kolejki. Niektóre atrybuty kolejki mają zastosowanie do wszystkich typów kolejek; inne atrybuty kolejki mają zastosowanie tylko do niektórych typów kolejek.

Typy kolejek

Menedżer kolejek obsługuje następujące typy definicji kolejek:

Kolejka lokalna
Komunikaty można przechowywać w kolejce lokalnej.

z/OS W systemie z/OS można ustawić ją jako kolejkę współużytkowaną lub prywatną.

Kolejka jest znana w programie jako *lokalna*, jeśli jej właścicielem jest menedżer kolejek, z którym połączony jest program. Komunikaty można pobierać z kolejek lokalnych oraz umieszczać je w nich.

Obiekt definicji kolejki przechowuje informacje o definicji kolejki, a także komunikaty fizyczne umieszczone w kolejce.

Kolejka lokalnego menedżera kolejek
Kolejka istnieje w lokalnym menedżerze kolejek.

z/OS Kolejka jest nazywana kolejką prywatną w systemie z/OS.

z/OS **Kolejka współużytkowana (tylko systemz/OS)**
Kolejka istnieje we współużytkowanym repozytorium dostępnym dla wszystkich menedżerów kolejek należących do grupy współużytkowania kolejek, do której należy repozytorium współużytkowane.

Aplikacje połączone z dowolnym menedżerem kolejek w grupie współużytkowania kolejek mogą umieszczać komunikaty w kolejkach tego typu i usuwać je z nich. Takie kolejki są w rzeczywistości takie same, jak kolejki lokalne. Wartością atrybutu kolejki **QType** jest MQQT_LOCAL.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu i usuwać je z nich. Wartością atrybutu kolejki **QType** jest MQQT_LOCAL.

Kolejka klastra

Komunikaty można przechowywać w kolejce klastra w menedżerze kolejek, w którym są zdefiniowane. Kolejka klastra to kolejka udostępniana przez menedżer kolejek klastra innym menedżerom kolejek w klastrze. Wartość atrybutu kolejki **QType** to MQQT_CLUSTER.

Definicja kolejki klastra jest ogłaszana w innych menedżerach kolejek w klastrze. Inne menedżery kolejek w klastrze mogą umieszczać komunikaty w kolejce klastra bez konieczności stosowania odpowiadającej jej definicji kolejki zdalnej. Kolejka klastra może zostać ogłoszona w więcej niż jednym klastrze przy użyciu listy nazw klastra.

Po ogłoszeniu kolejki każdy menedżer kolejek w klastrze może umieszczać w niej komunikaty. Aby umieścić komunikat, menedżer kolejek musi ustalić, w którym repozytorium pełnym znajduje się kolejka. Następnie do komunikatu w kolejce transmisji klastra dodawane są niektóre informacje o kierowaniu.

Menedżer kolejek może przechowywać komunikaty dla innych menedżerów kolejek w klastrze w wielu kolejkach transmisji. Menedżer kolejek można skonfigurować na dwa sposoby w celu przechowywania komunikatów w wielu kolejkach transmisji klastra. Jeśli atrybut menedżera kolejek **DEFCLXQ** zostanie ustawiony na wartość CHANNEL, inna kolejka transmisji klastra zostanie utworzona automatycznie z programu SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE dla każdego kanału nadawczego klastra. Jeśli opcja kolejki transmisji CLCHNAME zostanie ustawiona w taki sposób, aby była zgodna z co najmniej jednym kanałem nadawczym klastra, menedżer kolejek może przechowywać komunikaty dla zgodnych kanałów w kolejce transmisji.

Ostrzeżenie: Jeśli dedykowany produkt SYSTEM.CLUSTER.TRANSMIT.QUEUES jest używany z menedżerem kolejek, który został zaktualizowany z wersji produktu wcześniejszej niż IBM WebSphere MQ 7.5, należy upewnić się, że dla SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE opcję SHARE/NOSHARE ustawiono na wartość **SHARE**.

z/OS Kolejka klastra może być kolejką współużytkowaną przez członków grupy współużytkowania kolejki w programie IBM MQ for z/OS.

Kolejka zdalna
Kolejka zdalna nie jest kolejką fizyczną; jest lokalną definicją kolejki, która istnieje w zdalnym menedżerze kolejek. Definicja lokalna kolejki zdalnej zawiera informacje, które informują menedżera kolejek lokalnych o sposobie kierowania komunikatów do menedżera kolejek zdalnych.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu; komunikaty są umieszczane w lokalnej kolejce transmisji używanej do kierowania komunikatów do zdalnego menedżera kolejek. Aplikacje nie mogą usuwać komunikatów z kolejek zdalnych. Wartością atrybutu kolejki **QType** jest MQQT_REMOTE.

Definicji kolejki zdalnej można również użyć do:

- Używanie aliasów w kolejkach odpowiedzi

  W tym przypadku nazwa definicji jest nazwą kolejki odpowiedzi. Więcej informacji na ten temat zawiera sekcja Aliasy kolejek zwrotnych i klastry.

- Używanie aliasów menedżera kolejek

  W tym przypadku nazwa definicji jest aliasem menedżera kolejek, a nie nazwą kolejki. Więcej informacji na ten temat zawiera sekcja Aliasy i klastry menedżera kolejek.

Kolejka aliasowa
Nie jest to kolejka fizyczna; jest to nazwa alternatywna dla kolejki lokalnej, kolejki współużytkowanej, kolejki klastra lub kolejki zdalnej. Nazwa kolejki, do której alias jest tłumaczony, jest częścią definicji kolejki aliasowej.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu; komunikaty są umieszczane w kolejce, do której alias jest tłumaczony. Aplikacje mogą usuwać komunikaty z kolejek tego typu, jeśli alias jest tłumaczony na kolejkę lokalną, kolejkę współużytkowaną lub kolejkę klastra z instancją lokalną. Wartością atrybutu kolejki **QType** jest MQQT_ALIAS.

Kolejka modelowa
Nie jest to kolejka fizyczna; jest to zestaw atrybutów kolejki, na podstawie których można utworzyć kolejkę lokalną.

Komunikaty nie mogą być przechowywane w kolejkach tego typu.

limity kolejek

W produkcie IBM MQ 9.2.0dostępna jest opcja konfigurowania i monitorowania kolejek, które będą obsługiwać znacznie więcej niż domyślny limit dwóch terabajtów używany we wcześniejszych wersjach produktu IBM MQ. Dostępna jest również opcja zmniejszenia wielkości, do której może zostać powiększony plik kolejki.

Aby umożliwić konfigurowanie kolejek, można użyć atrybutu **MAXFSIZE** w kolejkach lokalnych i modelowych, a do monitorowania kolejek można użyć atrybutów statusu kolejki **CURFSIZE** i **CURMAXFS**.

Więcej informacji na ten temat zawiera sekcja Modyfikowanie plików kolejek systemu IBM MQ.

Kolejka - atrybuty

Niektóre atrybuty kolejki mają zastosowanie do wszystkich typów kolejek; inne atrybuty kolejki mają zastosowanie tylko do niektórych typów kolejek. Typy kolejek, do których atrybut ma zastosowanie, są przedstawione w tabeli Tabela 561 na stronie 865 i w kolejnych tabelach.

Tabela 561 na stronie 865 zawiera podsumowanie atrybutów specyficznych dla kolejek. Atrybuty są opisane w porządku alfabetycznym.

Uwaga: Nazwy atrybutów przedstawione w tej sekcji są opisowymi nazwami używanymi w wywołaniach MQINQ i MQSET ; nazwy są takie same, jak w przypadku komend PCF. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Szczegółowe informacje na ten temat zawiera sekcja Komendy MQSC .

W poniższej tabeli kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych dotyczy również kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można uzyskać, gdy kolejka klastra jest otwierana tylko dla zapytań lub dla zapytań i danych wyjściowych. Jeśli zostaną wyświetlone jakiekolwiek inne atrybuty, wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_SELECTOR_NOT_FOR_TYPE (2068).

  Jeśli kolejka klastra jest otwarta w celu wykonania zapytania plus co najmniej jedna z wartości wejściowych, przeglądania lub ustawiania, zamiast tego stosowana jest kolumna dla kolejek lokalnych.

  Jeśli kolejka klastra jest otwarta tylko na potrzeby wykonywania zapytań lub na potrzeby wykonywania zapytań i danych wyjściowych, a ponadto określono podstawową nazwę menedżera kolejek, zamiast niej stosowana jest kolumna dla kolejek lokalnych.

Tabela 561. Atrybuty kolejek

Atrybut	Opis	Lokalna	Model	Alias	Zdalna	Klaster
AlterationDate	Data ostatniej zmiany definicji	X		X	X	
AlterationTime	Czas ostatniej zmiany definicji	X		X	X	
BackoutRequeueQName	Nadmierna ilość kolejki wycofanych komunikatów	X	X			
BackoutThreshold	Próg wycofania	X	X			
BaseQName	Nazwa kolejki, na którą alias jest tłumaczony			X		
CFStrucName	Nazwa struktury narzędzia CF	X	X			
CLCHNAME	Nazwy kanałów nadawczych klastra	✓	✓			
ClusterName	Nazwa klastra, do którego należy kolejka	X		X	X	X
ClusterNamelist	Nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy kolejka	X		X	X	
CLWLQueuePriority	Priorytet kolejki obciążenia klastra	X		X	X	X
CLWLQueueRank	Klasyfikacja kolejki obciążenia klastra	X		X	X	X

Tabela 561. Atrybuty kolejek (kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalna	Klaster
CLWLUseQ	Użyj kolejki zdalnej	X				
CreationDate	Data utworzenia kolejki	X				
CreationTime	Czas utworzenia kolejki	X				
CurrentQDepth	Bieżąca głębokość kolejki	X				
DefaultPutResponse	Operacja put - domyślna odp.	✓	✓	✓	✓	
DefBind	Domyślne łączenie	X		X	X	X
DefinitionType attribute	Typ definicji kolejki.	X	X			
DefInputOpenOption	Domyślna opcja otwarcia wejścia	X	X			
DefPersistence	Domyślna trwałość komunikatu	X	X	X	X	X
DefPriority	Domyślny priorytet komunikatu	✓	✓	✓	✓	✓
DefReadAhead	Domyślny odczyt z wyprzedzeniem	X	X	X		
DistLists	Obsługa listy dystrybucyjnej	X	X			
HardenGetBackout	Czy zachować dokładną liczbę wycofań	X	X			
IndexType	Typ indeksu	X	X			
InhibitGet	Określa, czy operacje pobierania dla kolejki są dozwolone	X	X	X		
InhibitPut	Określa, czy dozwolone są operacje umieszczania (put) dla kolejki	X	X	X	X	X
InitiationQName	Nazwa kolejki inicjującej	X	X			
MaxMsgLength	Maksymalna długość komunikatu w bajtach	X	X			
MaxQDepth	Maksymalna głębokość kolejki	X	X			
MsgDeliverySequence attribute	Kolejność dostarczania komunikatów	X	X			
NonPersistentMessage Class	Cel niezawodności dla nietrwałych komunikatów	X	X			
OpenInputCount	Liczba operacji otwierania do wprowadzania	X				
OpenOutputCount	Liczba operacji otwarcia dla danych wyjściowych	X				
PropertyControl	Sterowanie właściwościami	✓	✓	✓		
ProcessName	Nazwa procesu	X	X			
QDepthHighEvent attribute	Określa, czy są generowane zdarzenia nadmiaru kolejki	X	X			
QDepthHighLimit	Górny limit głębokości kolejki	X	X			
QDepthLowEvent attribute	Informacja o tym, czy generowane są zdarzenia niedoboru kolejki	X	X			
QDepthLowLimit attribute	Dolny limit głębokości kolejki	X	X			
QDepthMaxEvent	Informacja o tym, czy są generowane zdarzenia zapełnienia kolejki	X	X			
QDesc	Opis kolejki	X	X	X	X	X

Tabela 561. Atrybuty kolejek (kontynuacja)						
Atrybut	**Opis**	**Lokalna**	**Model**	**Alias**	**Zdalna**	**Klaster**
QName	Nazwa kolejki	X		X	X	X
QServiceInterval	Miejsce docelowe dla odstępu czasu usługi kolejki	X	X			
QServiceIntervalEvent attribute	Określa, czy generowane są zdarzenia wysokiego, czy prawidłowego odstępu czasu usługi.	X	X			
QSGDisp attribute	Dyspozycja grupy współużytkowania kolejki	X		X	X	
QueueAccounting	Gromadzenie danych rozliczeniowych dla kolejki	X	X	X	X	X
QueueMonitoring	Dane monitorowania bezpośredniego dla kolejek	X	✓			
QueueStatistics	Gromadzenie danych statystycznych kolejki	X	X	X	X	X
QType	Typ kolejki	X		X	X	X
RemoteQMgrName	Nazwa zdalnego menedżera kolejek				X	
RemoteQName	Nazwa kolejki zdalnej				X	
RetentionInterval	Interwał przechowywania	X	X			
Scope	Określa, czy pozycja kolejki istnieje również w katalogu komórki	X		X	X	
Shareability	Możliwość współużytkowania kolejki	X	X			
StorageClass	Klasa pamięci dla kolejki	X	X			
TriggerControl	Kontrola wyzwalacza	X	X			
TriggerData	Dane wyzwalacza	X	X			
TriggerDepth	Wyzwalacz uruchamiany zapełnieniem	X	X			
TriggerMsgPriority	Próg priorytetu komunikatu dla wyzwalacza.	X	X			
TriggerType	Typ wyzwalacza	X	X			
Usage attribute	Wykorzystanie kolejki	X	X			
XmitQName	Nazwa kolejki transmisji				X	

Pojęcia pokrewne

Kolejki klastra

Kolejki lokalne

Jak wybrać typ kolejki transmisji klastra, który ma być używany

AlterationDate (MQCHAR12)

Data ostatniej zmiany definicji.

Tabela 562. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów (na przykład 1992-09-23¬¬, gdzie ¬¬ reprezentuje dwa znaki odstępu).

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) zmieniają się w miarę działania menedżera kolejek. Zmiany tych atrybutów nie mają wpływu na *AlterationDate*.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Czas ostatniej zmiany definicji.

Tabela 563. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS z użyciem zegara 24-godzinnego, z zerem wiodącym, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20).

- W systemie z/OSjest to czas Greenwich (GMT), przy czym zegar systemowy jest ustawiany dokładnie na czas GMT.
- W innych środowiskach czas jest czasem lokalnym.

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) zmieniają się w miarę działania menedżera kolejek. Zmiany tych atrybutów nie mają wpływu na *AlterationTime*.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_TIME_LENGTH.

BackoutRequeueNazwa QName (MQCHAR48)

Jest to nadmierna nazwa kolejki wycofanych komunikatów. Oprócz zezwolenia na wykonanie zapytania o jego wartość, menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość tego atrybutu.

Tabela 564. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Aplikacje działające w systemie WebSphere Application Server oraz aplikacje korzystające z narzędzi IBM MQ Application Server Facilities używają tego atrybutu do określenia miejsca, w którym powinny być wyświetlane wycofane komunikaty. W przypadku wszystkich innych aplikacji menedżer kolejek nie wykonuje żadnego działania w zależności od wartości atrybutu.

Program IBM MQ classes for JMS używa tego atrybutu do określenia miejsca, do którego ma zostać wysłany komunikat, który został już wycofany, maksymalną liczbę razy określoną przez atrybut *BackoutThreshold* .

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_BACKOUT_REQ_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

BackoutThreshold (MQLONG)

Jest to próg wycofania. Oprócz zezwolenia na wykonanie zapytania o jego wartość, menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość tego atrybutu.

Tabela 565. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Aplikacje działające w systemie WebSphere Application Server oraz aplikacje korzystające z narzędzia IBM MQ Application Server Facilities będą używać tego atrybutu do określenia, czy komunikat powinien zostać wycofany. W przypadku wszystkich innych aplikacji menedżer kolejek nie wykonuje żadnego działania w zależności od wartości atrybutu.

Program IBM MQ classes for JMS używa tego atrybutu do określenia, ile razy można zezwolić na wycofanie komunikatu przed przesłaniem go do kolejki określonej przez atrybut *BackoutRequeueQName* .

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_BACKOUT_THRESHOLD z wywołaniem MQINQ.

BaseQName (MQCHAR48)

Jest to nazwa kolejki, która jest zdefiniowana dla lokalnego menedżera kolejek.

Tabela 566. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
		X		

Więcej informacji na temat nazw kolejek zawiera sekcja MQOD-pole ObjectName. Kolejka może być jednego z następujących typów:

MQQT_LOCAL,
Kolejka lokalna.

MQQT_REMOTE
Lokalna definicja kolejki zdalnej.

MQQT_CLUSTER
Kolejka klastra.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_BASE_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

BaseType (MQCFIN)

Typ obiektu, na który alias jest tłumaczony.

Tabela 567. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
		X		

Jest to jedna z następujących wartości:

MQOT_Q
Podstawowym typem obiektu jest kolejka

TEMAT_MQOT
Podstawowy typ obiektu jest tematem

CFStrucName (MQCHAR12)
Jest to nazwa struktury narzędzia CF, w której przechowywane są komunikaty w kolejce. Pierwszy znak nazwy należy do zakresu od A do Z, a pozostałe znaki należą do zakresu od A do Z, od 0 do 9 lub są puste.

Tabela 568. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Aby uzyskać pełną nazwę struktury w narzędziu CF, należy dodać wartość atrybutu menedżera kolejek **QSGName** do przyrostka z wartością atrybutu kolejki **CFStrucName** .

Ten atrybut ma zastosowanie tylko do kolejek współużytkowanych. Jest on ignorowany, jeśli produkt *QSGDisp* nie ma wartości MQQSGD_SHARED.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CF_STRUC_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_CF_STRUC_NAME_LENGTH.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

ClusterChannelNazwa (MQCHAR20)
ClusterChannelNazwa to nazwa ogólna kanałów nadawczych klastra, które używają tej kolejki jako kolejki transmisji. Atrybut określa, które kanały nadawcze klastra wysyłają komunikaty do kanału odbiorczego klastra z tej kolejki transmisji klastra.

Tabela 569. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Domyślna konfiguracja menedżera kolejek dotyczy wszystkich kanałów nadawczych klastra mających wysyłać komunikaty z pojedynczej kolejki transmisji SYSTEM.CLUSTER.TRANSMIT.QUEUE. Konfigurację domyślną można zmienić, modyfikując, zmieniając atrybut menedżera kolejek **DefClusterXmitQueueType**. Wartością domyślną tego atrybutu jest SCTQ. Wartość tę można zmienić na CHANNEL. Jeśli atrybut **DefClusterXmitQueueType** zostanie ustawiony na wartość CHANNEL, dla każdego kanału nadawczego klastra domyślnie zostanie użyta konkretna kolejka transmisji klastra SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

Kanał nadawczy klastra dla atrybutu ClusterChannelName kolejki transmisji można również ustawić ręcznie. Komunikaty przeznaczone dla menedżera kolejek połączonego kanałem nadawczym klastra są przechowywane w kolejce transmisji identyfikującej kanał nadawczy klastra. Nie są one przechowywane w domyślnej kolejce transmisji klastra. Jeśli dla atrybutu ClusterChannelName zostaną ustawione wartości puste, po zrestartowaniu kanału zostanie on przełączony na domyślną kolejkę transmisji klastra. Kolejka domyślna to SYSTEM.CLUSTER.TRANSMIT.*ChannelName* lub SYSTEM.CLUSTER.TRANSMIT.QUEUE, w zależności od wartości atrybutu DefClusterXmitQueueType menedżera kolejek.

Określając gwiazdki ("*") w programie **ClusterChannelName**, można powiązać kolejkę transmisji z zestawem kanałów nadawczych klastra. Gwiazdki mogą znajdować się na początku, na końcu lub na dowolnej liczbie miejsc w środku łańcucha nazwy kanału. **ClusterChannelName** o długości ograniczonej do 20 znaków: MQ_CHANNEL_NAME_LENGTH.

ClusterName (MQCHAR48)
Jest to nazwa klastra, do którego należy kolejka.

Tabela 570. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Jeśli kolejka należy do więcej niż jednego klastra, parametr *ClusterNamelist* określa nazwę obiektu listy nazw, który identyfikuje klastry, a parametr *ClusterName* jest pusty. Co najmniej jedna z wartości *ClusterName* i *ClusterNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLUSTER_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_CLUSTER_NAME_LENGTH.

ClusterNamelist (MQCHAR48)

Jest to nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy ta kolejka.

Tabela 571. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	

Jeśli kolejka należy tylko do jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć parametru *ClusterName* , aby określić nazwę klastra, w którym to przypadku wartość *ClusterNamelist* jest pusta. Co najmniej jedna z wartości *ClusterName* i *ClusterNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CLUSTER_NAMELIST z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_NAMELIST_NAME_LENGTH.

CLWLQueuePriority (MQLONG),

Jest to priorytet kolejki obciążenia klastra, wartość z zakresu od 0 do 9 reprezentująca priorytet kolejki.

Tabela 572. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Więcej informacji na ten temat zawiera sekcja Kolejki klastrów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_Q_PRIORITY z wywołaniem MQINQ.

CLWLQueueRank (MQLONG)

Jest to pozycja kolejki obciążenia klastra, wartość z zakresu od 0 do 9 reprezentująca pozycję kolejki.

Tabela 573. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Więcej informacji na ten temat zawiera sekcja Kolejki klastrów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_Q_RANK z wywołaniem MQINQ.

CLWLUseQ (MQLONG),

Definiuje to zachowanie operacji MQPUT, gdy kolejka docelowa ma zarówno instancję lokalną, jak i co najmniej jedną instancję klastra zdalnego. Jeśli operacja put pochodzi z kanału klastra, ten atrybut nie ma zastosowania.

Tabela 574. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

Jest to jedna z następujących wartości:

MQCLWL_USEQ_ANY
Użyj kolejek zdalnych i lokalnych.

MQCLWL_USEQ_LOCAL.
Nie należy używać kolejek zdalnych.

MQCLWL_USEQ_AS_Q_MGR
Dziedzicz definicję z kolejki MQIA_CLWL_USEQ menedżera kolejek.

Więcej informacji na ten temat zawiera sekcja Kolejki klastrów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CLWL_USEQ z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_CLWL_USEQ_LENGTH.

CreationDate (MQCHAR12)
Data utworzenia kolejki.

Tabela 575. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu utworzenia długości 12 bajtów (na przykład 2013-09-23¬¬, gdzie ¬¬ reprezentuje 2 znaki odstępu).

- W systemie IBM idata utworzenia kolejki może różnić się od daty utworzenia bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CREATION_DATE z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_CREATION_DATE_LENGTH.

CreationTime (MQCHAR8)
Jest to czas utworzenia kolejki.

Tabela 576. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

Format godziny to HH.MM.SS z użyciem zegara 24-godzinnego, z zerem wiodącym, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20).

- W systemie z/OSjest to czas Greenwich (GMT), przy czym zegar systemowy jest ustawiany dokładnie na czas GMT.
- W innych środowiskach czas jest czasem lokalnym.
- W systemie IBM iczas utworzenia kolejki może różnić się od czasu utworzenia bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_CREATION_TIME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_CREATION_TIME_LENGTH.

CurrentQDepth (MQLONG)
To jest liczba komunikatów znajdujących się aktualnie w kolejce.

Tabela 577. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

Wartość ta jest zwiększana podczas wywołania MQPUT i wycofywania wywołania MQGET. Jest on zmniejszany podczas wykonywania wywołania MQGET bez przeglądania i podczas wycofywania

wywołania MQPUT. W rezultacie licznik uwzględnia komunikaty, które zostały umieszczone w kolejce w ramach jednostki pracy, ale nie zostały jeszcze zatwierdzone, nawet jeśli nie zostały zakwalifikowane do pobrania przez wywołanie MQGET. Podobnie wyklucza komunikaty, które zostały pobrane w ramach jednostki pracy za pomocą wywołania MQGET, ale które nie zostały jeszcze zatwierdzone.

Liczba ta obejmuje również komunikaty, które przekroczyli swój czas ważności, ale nie zostały jeszcze odrzucone, chociaż nie są zakwalifikowane do pobrania. Więcej informacji na ten temat zawiera sekcja MQMD-Expiry field .

Zarówno przetwarzanie jednostek pracy, jak i segmentacja komunikatów mogą spowodować przekroczenie przez *CurrentQDepth* wartości *MaxQDepth*. Nie ma to jednak wpływu na możliwość pobierania komunikatów. *wszystkie* komunikaty w kolejce można pobrać przy użyciu wywołania MQGET w normalny sposób.

Wartość tego atrybutu zmienia się w miarę działania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_CURRENT_Q_DEPTH z wywołaniem MQINQ.

Odpowiedź DefaultPut(MQLONG)

Określa typ odpowiedzi, która ma być używana dla operacji umieszczania w kolejce, gdy aplikacja określa opcję MQPMO_RESPONSE_AS_Q_DEF.

Tabela 578. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	

Jest to jedna z następujących wartości:

ODPOWIEDŹ MQPRT_SYNC_RESPONSE
Operacja put jest wykonywana synchronicznie i zwraca odpowiedź.

MQPRT_ASYNC_RESPONSE
Operacja put jest wykonywana asynchronicznie, zwracając podzbiór pól MQMD.

DefBind (MQLONG)

Jest to powiązanie domyślne używane, gdy w wywołaniu MQOPEN określono opcję MQOO_BIND_AS_Q_DEF, a kolejka jest kolejką klastra.

Tabela 579. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Jest to jedna z następujących wartości:

MQBND_BIND_ON_OPEN
Powiązanie naprawione przez wywołanie MQOPEN.

MQBND_BIND_NOT_FIXED (NIEUSTALONY)
Powiązanie nie jest stałe.

MQBND_BIND_ON_GROUP
Umożliwia aplikacji żądanie przydzielenia grupy komunikatów do tej samej instancji docelowej.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_BIND z wywołaniem MQINQ.

DefinitionType (MQLONG)

Wskazuje sposób zdefiniowania kolejki.

Tabela 580. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to jedna z następujących wartości:

MQQDT_PREDEFINED (predefiniowana)
Kolejka jest kolejką trwałą utworzoną przez administratora systemu; tylko administrator systemu może ją usunąć.

Predefiniowane kolejki są tworzone za pomocą komendy DEFINE MQSC i można je usunąć tylko za pomocą komendy DELETE MQSC. Nie można tworzyć predefiniowanych kolejek na podstawie kolejek modelowych.

Komendy mogą być wydawane przez operatora lub przez autoryzowanego użytkownika wysyłającego komunikat komendy do kolejki wejściowej komend (więcej informacji na ten temat zawiera opis atrybutu QName CommandInput).

MQQDT_PERMANENT_DYNAMIC,
Kolejka jest kolejką trwałą, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określonej w deskryptorze obiektu MQOD. Definicja kolejki modelowej miała wartość MQQDT_PERMANENT_DYNAMIC dla atrybutu **DefinitionType** .

Ten typ kolejki można usunąć za pomocą wywołania MQCLOSE. Więcej informacji na temat zawiera sekcja "MQCLOSE-zamknięcie obiektu" na stronie 669.

Wartością atrybutu **QSGDisp** dla trwałej kolejki dynamicznej jest MQQSGD_Q_MGR.

MQQDT_TEMPORARY_DYNAMIC
Kolejka jest kolejką tymczasową, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskryptorze obiektu MQOD. Definicja kolejki modelowej miała wartość MQQDT_TEMPORARY_DYNAMIC dla atrybutu **DefinitionType** .

Ten typ kolejki jest automatycznie usuwany przez wywołanie MQCLOSE po jego zamknięciu przez aplikację, która go utworzyła.

Wartością atrybutu **QSGDisp** dla tymczasowej kolejki dynamicznej jest MQQSGD_Q_MGR.

MQQDT_SHARED_DYNAMIC,
Kolejka jest współużytkowaną kolejką trwałą, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskryptorze obiektu MQOD. Definicja kolejki modelowej miała wartość MQQDT_SHARED_DYNAMIC dla atrybutu **DefinitionType** .

Ten typ kolejki można usunąć za pomocą wywołania MQCLOSE. Więcej informacji na temat zawiera sekcja "MQCLOSE-zamknięcie obiektu" na stronie 669.

Wartością atrybutu **QSGDisp** dla współużytkowanej kolejki dynamicznej jest MQQSGD_SHARED.

Ten atrybut w definicji kolejki modelowej nie wskazuje sposobu zdefiniowania kolejki modelowej, ponieważ kolejki modelowe są zawsze predefiniowane. Zamiast tego wartość tego atrybutu w kolejce modelowej jest używana do określenia wartości *DefinitionType* dla każdej kolejki dynamicznej utworzonej na podstawie definicji kolejki modelowej przy użyciu wywołania MQOPEN.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEFINITION_TYPE z wywołaniem MQINQ.

DefInputOpenOption (MQLONG)

Jest to domyślny sposób otwierania kolejki dla wejścia.

Tabela 581. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Ma ona zastosowanie, jeśli podczas otwierania kolejki w wywołaniu MQOPEN podano opcję MQOO_INPUT_AS_Q_DEF. Jest to jedna z następujących wartości:

MQOO_INPUT_EXCLUSIVE (wyłączna)
Otwórz kolejkę, aby pobrać komunikaty z dostępem na wyłączność.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację dla wejścia dowolnego typu (MQOO_INPUT_SHARED lub MQOO_INPUT_EXCLUSIVE).

MQOO_INPUT_SHARED
Otwórz kolejkę, aby pobrać komunikaty z dostępem współużytkowanym.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się powodzeniem, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację z opcją MQOO_INPUT_SHARED, ale kończy się niepowodzeniem z kodem przyczyny MQRC_OBJECT_IN_USE, jeśli kolejka jest obecnie otwarta z opcją MQOO_INPUT_EXCLUSIVE.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_INPUT_OPEN_OPTION z wywołaniem MQINQ.

DefPersistence (MQLONG) (DefPersistence)

Jest to domyślna trwałość komunikatów w kolejce. Ma ona zastosowanie, jeśli podczas umieszczania komunikatu w deskryptorze komunikatu określono parametr MQPER_PERSISTENCE_AS_Q_DEF.

Tabela 582. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w *pierwszej* definicji w ścieżce w czasie wywołania MQPUT lub MQPUT1 . Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład *DefXmitQName*)

Jest to jedna z następujących wartości:

MQPER_PERSISTENT
Komunikat przetrwa awarie systemu i restarty menedżera kolejek. Nie można umieścić trwałych komunikatów w:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane, które są odwzorowywane na obiekt CFSTRUCT w CFLEVEL (2) lub poniżej, lub w których obiekt CFSTRUCT jest zdefiniowany jako RECOVER (NO).

Komunikaty trwałe mogą być umieszczane w trwałych kolejkach dynamicznych oraz w predefiniowanych kolejkach.

MQPER_NOT_PERSISTENT
Komunikat zwykle nie przetrwa awarii systemu lub restartów menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartowania menedżera kolejek w pamięci dyskowej zostanie znaleziona nienaruszona kopia komunikatu.

W przypadku kolejek współużytkowanych komunikaty nietrwałe *pozostają* po restarcie menedżerów kolejek w grupie współużytkowania kolejek, ale nie są zachowywane awarie narzędzia CF używanego do przechowywania komunikatów we współużytkowanych kolejkach.

W tej samej kolejce mogą istnieć zarówno komunikaty trwałe, jak i nietrwałe.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_PERSISTENCE z wywołaniem MQINQ.

DefPriority (MQLONG)

Jest to domyślny priorytet komunikatów w kolejce. Ma to zastosowanie, jeśli parametr MQPRI_PRIORITY_AS_Q_DEF jest określony w deskryptorze komunikatu, gdy komunikat jest umieszczany w kolejce.

Tabela 583. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślny priorytet dla komunikatu jest pobierany z wartości tego atrybutu w *pierwszej* definicji w ścieżce w czasie operacji umieszczania (put). Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład *DefXmitQName*)

Sposób umieszczania komunikatu w kolejce zależy od wartości atrybutu **MsgDeliverySequence** kolejki:

- Jeśli atrybut **MsgDeliverySequence** ma wartość MQMDS_PRIORITY, pozycja logiczna, na której komunikat jest umieszczany w kolejce, zależy od wartości pola *Priority* w deskryptorze komunikatu.
- Jeśli atrybut **MsgDeliverySequence** ma wartość MQMDS_FIFO, komunikaty są umieszczane w kolejce tak, jakby miały priorytet równy *DefPriority* rozstrzygniętej kolejki, niezależnie od wartości pola *Priority* w deskryptorze komunikatu. Jednak pole *Priority* zachowuje wartość określoną przez aplikację, która umieściła komunikat. Więcej informacji na ten temat zawiera sekcja Atrybut sekwencjiMsgDelivery .

Priorytety należą do zakresu od 0 (najniższy) do *MaxPriority* (najwyższy); patrz atrybut MaxPriority.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_PRIORITY z wywołaniem MQINQ.

DefReadNieobecny (MQLONG)

Określa domyślne zachowanie odczytu z wyprzedzeniem dla nietrwałych komunikatów dostarczanych do klienta.

Tabela 584. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X		

DefReadMożna ustawić jedną z następujących wartości:

MQREADA_NO
Komunikaty nietrwałe nie są wysyłane do klienta przed zażądaniem ich przez aplikacje. Jeśli działanie klienta zostanie zakończone nieprawidłowo, może zostać utracony maksymalnie jeden komunikat nietrwały.

MQREADA_YES
Nietrwałe komunikaty są wysyłane do klienta przed zażądaniem ich przez aplikację. Nietrwałe komunikaty mogą zostać utracone, jeśli klient zostanie zakończony nieprawidłowo lub jeśli nie odbierze wszystkich wysłanych komunikatów.

MQREADA_WYŁĄCZONA
Odczyt z wyprzedzeniem nietrwałych komunikatów nie jest włączony dla tej kolejki. Komunikaty nie są wysyłane do klienta z wyprzedzeniem, niezależnie od tego, czy aplikacja kliencka żąda odczytu z wyprzedzeniem.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_READ_AHEAD z wywołaniem MQINQ.

DefPResp (MQLONG)

Domyślny atrybut typu odpowiedzi umieszczania (DEFPRESP) definiuje wartość używaną przez aplikacje, gdy dla typu PutResponsew MQPMO ustawiono wartość MQPMO_RESPONSE_AS_Q_DEF. Ten atrybut jest poprawny dla wszystkich typów kolejek.

Tabela 585. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jest to jedna z następujących wartości:

SYNCHRONICZNY
Operacja put jest wykonywana synchronicznie i zwraca odpowiedź.

ASYNCHRONICZNY
Operacja put jest wykonywana asynchronicznie, zwracając podzbiór pól MQMD.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DEF_PUT_RESPONSE_TYPE z wywołaniem MQINQ.

DistLists (MQLONG)

Wskazuje, czy komunikaty listy dystrybucyjnej mogą być umieszczane w kolejce.

Tabela 586. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Agent kanału komunikatów (MCA) ustawia atrybut w celu poinformowania lokalnego menedżera kolejek, czy menedżer kolejek na drugim końcu kanału obsługuje listy dystrybucyjne. Ten drugi menedżer kolejek (nazywany *partnerskim* menedżerem kolejek) odbiera komunikat po usunięciu go z lokalnej kolejki transmisji przez wysyłający agent MCA.

Wysyłający agent MCA ustawia atrybut za każdym razem, gdy nawiązuje połączenie z odbierającym agentem MCA w partnerskim menedżerze kolejek. W ten sposób wysyłający agent MCA może spowodować, że menedżer kolejek lokalnych umieści w kolejce transmisji tylko te komunikaty, które mogą zostać poprawnie przetworzone przez współpracujący menedżer kolejek.

Ten atrybut jest przeznaczony przede wszystkim do użytku z kolejkami transmisji, ale opisane przetwarzanie jest wykonywane niezależnie od użycia zdefiniowanego dla kolejki (patrz sekcja Atrybut użycia).

Jest to jedna z następujących wartości:

MQDL_SUPPORTED (obsługiwane)
Komunikaty z listy dystrybucyjnej mogą być przechowywane w kolejce i przesyłane do partnerskiej kolejki menedżera kolejek w tej postaci. Zmniejsza to ilość przetwarzania wymaganego do wysłania komunikatu do wielu miejsc docelowych.

NIEOBSŁUGIWANA MQDL_NOT_SUPPORTED
Komunikaty listy dystrybucyjnej nie mogą być składowane w kolejce, ponieważ menedżer kolejek partnerskich nie obsługuje list dystrybucyjnych. Jeśli aplikacja umieszcza komunikat listy dystrybucyjnej i komunikat ten ma zostać umieszczony w tej kolejce, menedżer kolejek dzieli komunikat listy dystrybucyjnej i umieszcza zamiast tego poszczególne komunikaty w kolejce. Zwiększa to ilość przetwarzania wymaganego do wysłania komunikatu do wielu miejsc docelowych, ale zapewnia poprawne przetwarzanie komunikatów przez partnerujący menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_DIST_LISTS z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

Ten atrybut nie jest obsługiwany w systemie z/OS.

Wycofania HardenGet(MQLONG)

Dla każdego komunikatu jest zachowywana liczba określająca, ile razy komunikat został pobrany przez wywołanie MQGET w obrębie jednostki pracy, a następnie wycofany.

Tabela 587. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Ten licznik jest dostępny w polu *BackoutCount* w deskryptorze komunikatu po zakończeniu wywołania MQGET.

Liczba wycofanych komunikatów przetrwa restarty menedżera kolejek. Aby jednak zapewnić dokładność licznika, informacje muszą być *wzmocnione* (zapisane na dysku lub innym trwałym urządzeniu pamięci masowej) za każdym razem, gdy wywołanie MQGET pobiera komunikat w ramach jednostki pracy dla tej kolejki. Jeśli ta czynność nie zostanie wykonana, działanie menedżera kolejek zakończy się niepowodzeniem, a wywołania MQGET zostaną zwrócone, licznik może, ale nie musi, zostać zwiększony.

Wzmocnienie informacji dla każdego wywołania MQGET w obrębie jednostki pracy powoduje jednak dodatkowy koszt przetwarzania, dlatego atrybut **HardenGetBackout** należy ustawić na wartość MQQA_BACKOUT_HARTOWANE tylko wtedy, gdy istotne jest, aby licznik był dokładny.

W systemie Wiele platformliczba wycofanych komunikatów jest zawsze zachowana, niezależnie od ustawienia tego atrybutu.

Dozwolone są następujące wartości:

MQQA_BACKOUT_HARTOWANE
Wzmacnianie jest używane w celu zapewnienia, że liczba wycofanych komunikatów w tej kolejce jest dokładna.

MQQA_BACKOUT_NOT_HARTOWANE
Wzmacnianie nie jest używane w celu upewnienia się, że liczba wycofanych komunikatów w tej kolejce jest dokładna. W związku z tym liczba ta może być niższa niż powinna.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_HARDEN_GET_BACKOUT z wywołaniem MQINQ.

IndexType (MQLONG)

Określa typ indeksu obsługiwanego przez menedżer kolejek dla komunikatów w kolejce.

Tabela 588. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Typ wymaganego indeksu zależy od sposobu, w jaki aplikacja pobiera komunikaty, oraz od tego, czy kolejka jest kolejką współużytkowaną, czy niewspółużytkowaną (patrz atrybut QSGDisp). *IndexType*może przyjmować następujące wartości:

MQIT_BRAK
Żaden indeks nie jest obsługiwany przez menedżera kolejek dla tej kolejki. Tej wartości należy użyć dla kolejek, które są zwykle przetwarzane sekwencyjnie, tzn. bez użycia kryteriów wyboru w wywołaniu MQGET.

MQIT_MSG_ID
Menedżer kolejek obsługuje indeks, który używa identyfikatorów komunikatów w kolejce. Tej wartości należy użyć w kolejkach, w których aplikacja zwykle pobiera komunikaty przy użyciu identyfikatora komunikatu jako kryterium wyboru w wywołaniu MQGET.

MQIT_CORREL_ID (Identyfikator KORELACJI MQ)
Menedżer kolejek obsługuje indeks, który używa identyfikatorów korelacji komunikatów w kolejce. Tej wartości należy użyć w przypadku kolejek, w których aplikacja zwykle pobiera komunikaty przy użyciu identyfikatora korelacji jako kryterium wyboru w wywołaniu MQGET.

MQIT_MSG_TOKEN,

Ważne: Ten typ indeksu powinien być używany tylko dla kolejek używanych z produktem IBM MQ Workflow for z/OS .

Menedżer kolejek przechowuje indeks, który używa znaczników komunikatów w kolejce do użycia z funkcjami menedżera obciążenia (WLM) programu z/OS.

Ta opcja *musi* być określona dla kolejek zarządzanych przez WLM. Nie należy jej określać dla żadnego innego typu kolejek. Nie należy również używać tej wartości w przypadku kolejki, w której aplikacja nie używa funkcji menedżera obciążenia z/OS , ale pobiera komunikaty przy użyciu znacznika komunikatu jako kryterium wyboru w wywołaniu MQGET.

MQIT_GROUP_ID (identyfikator grupy MQT)
Menedżer kolejek obsługuje indeks, który używa identyfikatorów grup komunikatów w kolejce. Ta wartość musi być używana dla kolejek, w których aplikacja pobiera komunikaty przy użyciu opcji MQGMO_LOGICAL_ORDER w wywołaniu MQGET.

Kolejka o tym typie indeksu nie może być kolejką transmisji. Kolejka współużytkowana o tym typie indeksu musi być zdefiniowana w celu odwzorowania na obiekt CFSTRUCT na poziomie CFLEVEL (3) lub wyższym.

Uwaga:

1. Porządek fizyczny komunikatów w kolejce o typie indeksu MQIT_GROUP_ID nie został zdefiniowany, ponieważ kolejka jest zoptymalizowana pod kątem wydajnego pobierania komunikatów przy użyciu opcji MQGMO_LOGICAL_ORDER w wywołaniu MQGET. Oznacza to, że fizyczna kolejność komunikatów nie jest zwykle kolejnością, w której komunikaty dotarły do kolejki.
2. Jeśli kolejka MQIT_GROUP_ID ma atrybut *MsgDeliverySequence* o wartości MQMDS_PRIORITY, menedżer kolejek używa priorytetów komunikatów 0 i 1 w celu zoptymalizowania pobierania komunikatów w porządku logicznym. W związku z tym pierwszy komunikat w grupie nie może mieć priorytetu zero lub jeden. Jeśli tak, komunikat jest przetwarzany tak, jakby miał priorytet dwa. Pole *Priority* w strukturze MQMD nie jest zmieniane.

Więcej informacji na temat grup komunikatów zawiera opis opcji grupy i segmentu w polu MQGMO-Opcje.

Typ indeksu, który powinien być używany w różnych przypadkach, jest przedstawiony w sekcji Tabela 589 na stronie 879 i Tabela 590 na stronie 881.

Tabela 589. Sugerowane lub wymagane wartości typu indeksu kolejki, gdy nie określono parametru MQGMO_LOGICAL_ORDER

Kryteria wyboru w wywołaniu MQGET	**Typ indeksu dla kolejki niewspółużytkowanej**	**Typ indeksu dla kolejki współużytkowanej**
Brak	Dowolna	Dowolna

Tabela 589. Sugerowane lub wymagane wartości typu indeksu kolejki, gdy nie określono parametru MQGMO_LOGICAL_ORDER (kontynuacja)

Kryteria wyboru w wywołaniu MQGET	**Typ indeksu dla kolejki niewspółużytkowanej**	**Typ indeksu dla kolejki współużytkowanej**
Wybór przy użyciu jednego identyfikatora:		
Identyfikator komunikatu	Sugerowany identyfikator MQIT_MSG_ID	Wymagane MQIT_NONE lub MQIT_MSG_ID; sugerowany MQIT_MSG_ID
Identyfikator korelacji	Sugerowany identyfikator MQIT_CORREL_ID	Wymagany identyfikator MQIT_CORREL_ID
Identyfikator grupy	MQIT_GROUP_ID sugerowany	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu dwóch identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji	Sugerowane MQIT_MSG_ID lub MQIT_CORREL_ID	Wymagane MQIT_NONE lub MQIT_MSG_ID lub MQIT_CORREL_ID (W celu zwiększenia efektywności zaleca się, aby typ indeksu był zgodny z polem MQMD, które będzie miało najwięcej odrębnych kluczy).
Identyfikator komunikatu plus identyfikator grupy	MQIT_MSG_ID lub MQIT_GROUP_ID sugerowane	Nieobsługiwane
Identyfikator korelacji plus identyfikator grupy	MQIT_CORREL_ID lub MQIT_GROUP_ID sugerowane	Nieobsługiwane
Wybór przy użyciu trzech identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji plus identyfikator grupy	MQIT_MSG_ID lub MQIT_CORREL_ID lub MQIT_GROUP_ID sugerowane	Nieobsługiwane
Wybór przy użyciu kryteriów powiązanych z grupą:		
Identyfikator grupy plus numer kolejny komunikatu	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Numer kolejny komunikatu (musi być 1)	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu znacznika komunikatu:		
Znacznik komunikatu do użycia przez aplikację	Nie używaj znacznika MQIT_MSG_TOKEN	
Znacznik komunikatu do użycia przez WLM	Wymagany MQIT_MSG_TOKEN	Nieobsługiwane

Tabela 590. Sugerowane lub wymagane wartości typu indeksu kolejki, gdy określono parametr MQGMO_LOGICAL_ORDER		
Kryteria wyboru w wywołaniu MQGET	**Typ indeksu dla kolejki niewspółużytkowanej**	**Typ indeksu dla kolejki współużytkowanej**
Brak	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu jednego identyfikatora:		
Identyfikator komunikatu	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator korelacji	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Wymagana wartość MQIT_GROUP_ID
Wybór przy użyciu dwóch identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator komunikatu plus identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Identyfikator korelacji plus identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane
Wybór przy użyciu trzech identyfikatorów:		
Identyfikator komunikatu plus identyfikator korelacji plus identyfikator grupy	Wymagana wartość MQIT_GROUP_ID	Nieobsługiwane

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INDEX_TYPE z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

InhibitGet (MQLONG)

Określa, czy operacje pobierania dla tej kolejki są dozwolone.

Tabela 591. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X	X		

Jeśli kolejka jest kolejką aliasową, operacje pobierania muszą być dozwolone zarówno dla kolejki aliasowej, jak i dla kolejki podstawowej w czasie wykonywania operacji pobierania, aby wywołanie MQGET powiodło się. Jest to jedna z następujących wartości:

MQQA_GET_INHIBITED
Operacje pobierania są zablokowane.

Wywołania MQGET nie powiodły się z kodem przyczyny MQRC_GET_INHIBITED. Obejmuje to wywołania MQGET, które określają MQGMO_BROWSE_FIRST lub MQGMO_BROWSE_NEXT.

Uwaga: Jeśli wywołanie MQGET działające w obrębie jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu **InhibitGet** na MQQA_GET_INHIBITED nie uniemożliwi zatwierdzenia jednostki pracy.

MQQA_GET_ALLOWED
Operacje pobierania są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INHIBIT_GET z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

InhibitPut (MQLONG)
Określa, czy operacje umieszczania dla tej kolejki są dozwolone.

Tabela 592. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, operacje umieszczania muszą być dozwolone dla *każdej* definicji w ścieżce (w tym dla wszystkich definicji aliasów menedżera kolejek) w czasie operacji umieszczania, aby wywołanie MQPUT lub MQPUT1 zakończyło się powodzeniem. Jest to jedna z następujących wartości:

MQQA_PUT_INHIBITED
Operacje umieszczania są zablokowane.

Wywołania MQPUT i MQPUT1 kończą się niepowodzeniem z kodem przyczyny MQRC_PUT_INHIBITED.

Uwaga: Jeśli wywołanie MQPUT działające w obrębie jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu **InhibitPut** na MQQA_PUT_INHIBITED nie uniemożliwi zatwierdzenia jednostki pracy.

MQQA_PUT_ALLOWED
Operacje umieszczania (put) są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_INHIBIT_PUT z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

InitiationQName (MQCHAR48)
Jest to nazwa kolejki zdefiniowana w lokalnym menedżerze kolejek. Kolejka musi być typu MQQT_LOCAL.

Tabela 593. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

Menedżer kolejek wysyła komunikat wyzwalacza do kolejki inicjującej, gdy w wyniku nadejścia komunikatu do kolejki, do której należy ten atrybut, wymagane jest uruchomienie aplikacji. Kolejka inicjowania musi być monitorowana przez aplikację monitora wyzwalacza, która uruchamia odpowiednią aplikację po odebraniu komunikatu wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_INITIATION_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

MaxMsg(MQLONG)
Jest to górny limit długości najdłuższego *fizycznego* komunikatu, który można umieścić w kolejce.

Tabela 594. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jednak ze względu na to, że atrybut kolejki **MaxMsgLength** można ustawić niezależnie od atrybutu menedżera kolejek systemu **MaxMsgLength** , rzeczywisty górny limit długości najdłuższego komunikatu fizycznego, który można umieścić w kolejce, jest mniejszy z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, aplikacja może umieścić w strukturze MQMD komunikat *logiczny* , który jest dłuższy niż mniejsza z dwóch atrybutów **MaxMsgLength** , ale tylko wtedy, gdy w aplikacji określono opcję MQMF_SEGMENTATION_ALLOWED. Jeśli ta opcja jest określona, górny limit długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zazwyczaj ograniczenia zasobów narzucane przez system operacyjny lub środowisko, w którym działa aplikacja, powodują ograniczenie dolnego limitu.

Próba umieszczenia w kolejce zbyt długiego komunikatu kończy się niepowodzeniem z jednym z następujących kodów przyczyny:

- MQRC_MSG_TOO_BIG_FOR_Q, jeśli komunikat jest zbyt duży dla kolejki
- MQRC_MSG_TOO_BIG_FOR_Q_MGR, jeśli komunikat jest zbyt duży dla menedżera kolejek, ale nie zbyt duży dla kolejki

Dolny limit dla atrybutu **MaxMsgLength** wynosi zero; górny limit wynosi 100 MB (104 857 600 bajtów).

Więcej informacji na ten temat zawiera sekcja MQPUT-parametr BufferLength.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_MSG_LENGTH z wywołaniem MQINQ.

MaxQDepth (MQLONG)

Jest to zdefiniowany górny limit liczby komunikatów fizycznych, które mogą istnieć w kolejce w dowolnym momencie.

Tabela 595. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Próba umieszczenia komunikatu w kolejce, która już zawiera komunikaty **MaxQDepth** , kończy się niepowodzeniem z kodem przyczyny MQRC_Q_FULL.

Zarówno przetwarzanie jednostki pracy, jak i segmentacja komunikatów mogą spowodować przekroczenie rzeczywistej liczby komunikatów fizycznych w kolejce **MaxQDepth**. Nie ma to jednak wpływu na możliwość pobierania komunikatu, ponieważ wszystkie komunikaty w kolejce można pobrać za pomocą wywołania MQGET.

Wartość tego atrybutu wynosi zero lub więcej. Górny limit jest określany przez środowisko:

- Na następujących platformach wartość nie może przekraczać 999 999 999:
  - AIX
  - Linux
  - Windows
  - z/OS
- IBM i W systemie IBM iwartość nie może przekraczać 640 000.

Uwaga: Przestrzeń pamięci dostępna dla kolejki może zostać wyczerpana, nawet jeśli w kolejce znajduje się mniej niż **MaxQDepth** komunikatów.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MAX_Q_DEPTH z wywołaniem MQINQ.

Sekwencja MsgDelivery(MQLONG)

Tabela 596. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Określa to kolejność, w jakiej wywołanie MQGET zwraca komunikaty do aplikacji:

MQMDS_FIFO,
Komunikaty są zwracane w kolejności FIFO (pierwszy przyszedł, pierwszy wyszedł).

Wywołanie MQGET zwraca *pierwszy* komunikat, który spełnia kryteria wyboru określone w wywołaniu, niezależnie od priorytetu komunikatu.

MQMDS_PRIORITY,
Komunikaty są zwracane w kolejności priorytetów.

Wywołanie MQGET zwraca komunikat *o najwyższym priorytecie* , który spełnia kryteria wyboru określone w wywołaniu. W ramach każdego poziomu priorytetu komunikaty są zwracane w kolejności FIFO (pierwszy przyszedł, pierwszy wyszedł).

- W systemie z/OS, jeśli kolejka ma atrybut *IndexType* o wartości MQIT_GROUP_ID, atrybut **MsgDeliverySequence** określa kolejność, w jakiej grupy komunikatów są zwracane do aplikacji. Konkretna sekwencja, w której grupy są zwracane, jest określana na podstawie pozycji lub priorytetu pierwszego komunikatu w każdej grupie. Kolejność fizyczna komunikatów w kolejce nie jest zdefiniowana, ponieważ kolejka jest zoptymalizowana pod kątem wydajnego pobierania komunikatów przy użyciu opcji MQGMO_LOGICAL_ORDER w wywołaniu MQGET.
- W systemie z/OS, jeśli parametr *IndexType* ma wartość MQIT_GROUP_ID, a parametr *MsgDeliverySequence* ma wartość MQMDS_PRIORITY, menedżer kolejek używa priorytetów komunikatów 0 i 1 w celu zoptymalizowania pobierania komunikatów w porządku logicznym. W związku z tym pierwszy komunikat w grupie nie może mieć priorytetu zero lub jeden. Jeśli tak, komunikat jest przetwarzany tak, jakby miał priorytet dwa. Pole *Priority* w strukturze MQMD nie jest zmieniane.

Jeśli odpowiednie atrybuty zostaną zmienione, gdy w kolejce znajdują się komunikaty, sekwencja dostarczania będzie następująca:

- Kolejność, w jakiej komunikaty są zwracane przez wywołanie MQGET, jest określana przez wartości atrybutów **MsgDeliverySequence** i **DefPriority** obowiązujących dla kolejki w momencie, gdy komunikat pojawia się w kolejce:
  - Jeśli w momencie nadejścia komunikatu parametr *MsgDeliverySequence* ma wartość MQMDS_FIFO, komunikat jest umieszczany w kolejce tak, jakby jego priorytet miał wartość *DefPriority*. Nie ma to wpływu na wartość pola *Priority* w deskryptorze komunikatu. To pole zachowuje wartość, którą miało podczas pierwszego umieszczania komunikatu.
  - Jeśli parametr *MsgDeliverySequence* ma wartość MQMDS_PRIORITY, gdy komunikat zostanie odebrany, komunikat jest umieszczany w kolejce w miejscu odpowiednim dla priorytetu podanego w polu *Priority* w deskryptorze komunikatu.

  Jeśli wartość atrybutu **MsgDeliverySequence** zostanie zmieniona, gdy w kolejce znajdują się komunikaty, kolejność komunikatów w kolejce nie zostanie zmieniona.

  Jeśli wartość atrybutu **DefPriority** zostanie zmieniona, gdy w kolejce znajdują się komunikaty, komunikaty nie muszą być dostarczane w kolejności FIFO, nawet jeśli atrybut **MsgDeliverySequence** jest ustawiony na wartość MQMDS_FIFO; te, które zostały umieszczone w kolejce z wyższym priorytetem, są dostarczane jako pierwsze.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MSG_DELIVERY_SEQUENCE z wywołaniem MQINQ.

NonPersistentMessageClass (MQLONG)
Cel niezawodności dla nietrwałych komunikatów.

Tabela 597. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Określa to okoliczności, w których nietrwałe komunikaty umieszczane w tej kolejce są usuwane:

MQNPM_CLASS_NORMAL
Nietrwałe komunikaty są ograniczone do czasu życia sesji menedżera kolejek. Komunikaty są usuwane w przypadku restartu menedżera kolejek. Wartość ta jest poprawna tylko dla kolejek niewspółużytkowanych i jest wartością domyślną.

MQNPM_CLASS_HIGH,
Menedżer kolejek próbuje zachować nietrwałe komunikaty przez czas życia kolejki. Komunikaty nietrwałe mogą zostać utracone w przypadku awarii. Ta wartość jest wymuszana dla kolejek współużytkowanych.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_NPM_CLASS z wywołaniem MQINQ.

Liczba operacji OpenInput(MQLONG)
Jest to liczba uchwytów, które są obecnie poprawne do usuwania komunikatów z kolejki za pomocą wywołania MQGET.

Tabela 598. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X				

Jest to łączna liczba takich uchwytów znanych *lokalnemu* menedżerowi kolejek. Jeśli kolejka jest kolejką współużytkowaną, licznik nie obejmuje operacji otwierania do wprowadzania, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych.

Licznik obejmuje uchwyty, w których kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta do wprowadzania. Licznik nie obejmuje uchwytów, w których otwarto kolejkę dla działań, które nie zawierały danych wejściowych (na przykład kolejka otwarta tylko do przeglądania).

Wartość tego atrybutu zmienia się w miarę działania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OPEN_INPUT_COUNT z wywołaniem MQINQ.

Liczba operacji OpenOutput(MQLONG)
Jest to liczba uchwytów, które są obecnie poprawne w przypadku dodawania komunikatów do kolejki za pomocą wywołania MQPUT.

Tabela 599. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X				

Jest to łączna liczba takich uchwytów znanych *lokalnemu* menedżerowi kolejek. Nie obejmuje to operacji otwierania do wyprowadzania wykonanych dla tej kolejki w zdalnych menedżerach kolejek. Jeśli kolejka jest kolejką współużytkowaną, liczba ta nie obejmuje otwartych danych wyjściowych, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych.

Liczba ta obejmuje uchwyty, w których kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla danych wyjściowych. Licznik nie obejmuje uchwytów, w których otwarto kolejkę dla działań, które nie zawierały danych wyjściowych (na przykład kolejka otwarta tylko na potrzeby zapytania).

Wartość tego atrybutu zmienia się w miarę działania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_OPEN_OUTPUT_COUNT z wywołaniem MQINQ.

ProcessName (MQCHAR48)

Jest to nazwa obiektu procesu, który jest zdefiniowany w lokalnym menedżerze kolejek. Obiekt procesu identyfikuje program, który może obsługiwać kolejkę.

Tabela 600. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_PROCESS_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_PROCESS_NAME_LENGTH.

PropertyControl (MQLONG)

Określa sposób obsługi właściwości komunikatów pobieranych z kolejek za pomocą wywołania MQGET z opcją MQGMO_PROPERTIES_AS_Q_DEF.

Tabela 601. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X		

Jest to jedna z następujących wartości:

MQPROP_ALL
Wszystkie właściwości komunikatu są dołączane do komunikatu, gdy jest on dostarczany do aplikacji. Właściwości te, z wyjątkiem tych, które znajdują się w deskryptorze komunikatu (lub rozszerzeniu), zostają umieszczone w jednym lub większej liczbie nagłówków MQRFH2 danych komunikatu. W przypadku dostarczenia uchwytu komunikatu zachowanie polega na zwróceniu właściwości w uchwycie komunikatu.

MQPROP_KOMPATYBILNOŚĆ
Jeśli komunikat zawiera właściwość z przedrostkiem mcd., jms, usr. lub mqext., wszystkie właściwości komunikatu są dostarczane do aplikacji w nagłówku MQRFH2 . W przeciwnym razie wszystkie właściwości komunikatu z wyjątkiem tych, które są zawarte w deskryptorze komunikatu lub w rozszerzeniu, są usuwane i nie są już dostępne dla aplikacji. Jest to wartość domyślna. Umożliwia ona aplikacjom, które oczekują, że właściwości powiązane z produktem JMS będą w nagłówku MQRFH2 danych komunikatu, kontynuowanie pracy bez modyfikacji. W przypadku dostarczenia uchwytu komunikatu zachowanie polega na zwróceniu właściwości w uchwycie komunikatu.

MQPROP_FORCE_MQRFH2
Właściwości są zawsze zwracane w danych komunikatu w nagłówku MQRFH2 (niezależnie od tego, czy aplikacja określa uchwyt komunikatu). Poprawny uchwyt komunikatu podany w polu MsgHandle struktury MQGMO w wywołaniu MQGET jest ignorowany. Właściwości komunikatu nie są dostępne poprzez uchwyt komunikatu.

MQPROP_NONE
Wszystkie właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskryptorze komunikatu (lub rozszerzeniu), są usuwane z komunikatu przed dostarczeniem komunikatu do aplikacji. W przypadku dostarczenia uchwytu komunikatu zachowanie polega na zwróceniu właściwości w uchwycie komunikatu.

Ten parametr ma zastosowanie do kolejek lokalnych, aliasowych i modelowych. Aby określić jego wartość, należy użyć selektora MQIA_PROPERTY_CONTROL z wywołaniem MQINQ.

Zdarzenie QDepthHigh(MQLONG)

Ta opcja określa, czy mają być generowane zdarzenia nadmiaru kolejki.

Tabela 602. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Zdarzenie Duże zapełnienie kolejki wskazuje, że aplikacja umieściła komunikat w kolejce, co spowodowało, że liczba komunikatów w kolejce stała się większa lub równa wysokiemu progowi zapełnienia kolejki (patrz atrybut **QDepthHighLimit**).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_HIGH_EVENT z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale nie można użyć wywołania MQINQ do określenia jego wartości.

QDepthHighLimit (MQLONG)

Jest to próg, z którym porównywane jest zapełnienie kolejki w celu wygenerowania zdarzenia dużego zapełnienia kolejki.

Tabela 603. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

To zdarzenie wskazuje, że aplikacja umieściła komunikat w kolejce, co spowodowało, że liczba komunikatów w kolejce stała się większa lub równa wysokiemu progowi zapełnienia kolejki. Patrz atrybut zdarzeniaQDepthHigh.

Wartość jest wyrażona jako procent maksymalnego zapełnienia kolejki (atrybut**MaxQDepth**) i jest większa lub równa 0 oraz mniejsza lub równa 100. Wartość domyślna to 80.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_HIGH_LIMIT z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale nie można użyć wywołania MQINQ do określenia jego wartości.

Zdarzenie QDepthLow(MQLONG)

Ta opcja określa, czy generowane są zdarzenia niedoboru kolejki.

Tabela 604. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Zdarzenie niedoboru kolejki wskazuje, że aplikacja pobrała komunikat z kolejki, co spowodowało, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnemu progowi zapełnienia kolejki (patrz atrybut QDepthLowLimit).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_LOW_EVENT z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale nie można użyć wywołania MQINQ do określenia jego wartości.

QDepthLowLimit (MQLONG)

Jest to próg, z którym porównywane jest zapełnienie kolejki w celu wygenerowania zdarzenia niedoboru kolejki.

Tabela 605. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

To zdarzenie wskazuje, że aplikacja pobrała komunikat z kolejki, co spowodowało, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnemu progowi zapełnienia kolejki. Patrz atrybut zdarzeniaQDepthLow.

Wartość jest wyrażona jako procent maksymalnego zapełnienia kolejki (atrybut**MaxQDepth**) i jest większa lub równa 0 oraz mniejsza lub równa 100. Wartością domyślną jest 20.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_LOW_LIMIT z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale nie można użyć wywołania MQINQ do określenia jego wartości.

Zdarzenie QDepthMax(MQLONG)

Ta opcja określa, czy mają być generowane zdarzenia zapełnienia kolejki. Zdarzenie zapełnienia kolejki wskazuje, że operacja umieszczenia w kolejce została odrzucona, ponieważ kolejka jest pełna, czyli zapełnienie kolejki osiągnęło już maksymalną wartość.

Tabela 606. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Jest to jedna z następujących wartości:

MQEVR_DISABLED (wyłączone)
Raportowanie zdarzeń jest wyłączone.

MQEVR_ENABLED,
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_DEPTH_MAX_EVENT z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale nie można użyć wywołania MQINQ do określenia jego wartości.

Opis kolejki (MQCHAR64)

To pole służy do opisowego komentarza.

Tabela 607. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole może zawierać znaki DBCS (z maksymalną długością pola wynoszącą 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_DESC z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_DESC_LENGTH.

QName (MQCHAR48)

Jest to nazwa kolejki zdefiniowana w menedżerze kolejek lokalnych.

Tabela 608. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Wszystkie kolejki zdefiniowane w menedżerze kolejek współużytkują tę samą przestrzeń nazw kolejki. Dlatego kolejki MQQT_LOCAL i MQQT_ALIAS nie mogą mieć takiej samej nazwy.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

QServiceInterval (MQLONG)

Jest to odstęp czasu usługi używany do porównywania w celu wygenerowania zdarzeń wysokiego i prawidłowego odstępu czasu usługi.

Tabela 609. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Patrz atrybut zdarzeniaQServiceInterval.

Wartość jest wyrażona w milisekundach i jest większa lub równa zero oraz mniejsza lub równa 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_SERVICE_INTERVAL z wywołaniem MQINQ.

Ten atrybut jest obsługiwany w systemie z/OS, ale nie można użyć wywołania MQINQ do określenia jego wartości.

Zdarzenie QServiceInterval(MQLONG)

Określa, czy generowane są zdarzenia wysokiego, czy też prawidłowego odstępu czasu usługi.

Tabela 610. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

- Zdarzenie wysokiego odstępu czasu usługi jest generowane, gdy sprawdzenie wskazuje, że nie pobrano żadnych komunikatów z kolejki przez co najmniej jeden czas wskazany przez atrybut **`QServiceInterval`** .
- Zdarzenie OK odstępu czasu usługi jest generowane, gdy sprawdzenie wskazuje, że komunikaty zostały pobrane z kolejki w czasie wskazanym przez atrybut **`QServiceInterval`** .

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Jest to jedna z następujących wartości:

MQQSIE_HIGH
Włączone zdarzenia wysokiego odstępu czasu usługi kolejki.

- Zdarzenia wysokiego odstępu czasu usługi kolejki są **włączone** i
- Zdarzenia OK odstępu czasu usługi kolejki są **wyłączone**.

MQQSIE_OK
Włączono zdarzenia OK odstępu czasu usługi kolejki.

- Zdarzenia wysokiego odstępu czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK odstępu czasu usługi kolejki są **włączone**.

MQQSIE_BRAK
Nie włączono żadnych zdarzeń odstępu czasu usługi kolejki.

- Zdarzenia wysokiego odstępu czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK odstępu czasu usługi kolejki są również **wyłączone**.

W przypadku kolejek współużytkowanych wartość tego atrybutu jest ignorowana; przyjmowana jest wartość MQQSIE_NONE.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_SERVICE_INTERVAL_EVENT z wywołaniem MQINQ.

W systemie z/OSnie można użyć wywołania MQINQ do określenia wartości tego atrybutu.

QSGDisp (MQLONG)

Określa dyspozycję kolejki.

Tabela 611. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	

Jest to jedna z następujących wartości:

MQQSGD_Q_MGR
Obiekt ma dyspozycję menedżera kolejek. Oznacza to, że definicja obiektu jest znana tylko menedżerowi kolejek lokalnych; definicja nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć obiekt o takiej samej nazwie i typie jak bieżący obiekt, ale są to oddzielne obiekty i nie ma między nimi korelacji. Ich atrybuty nie są ograniczone do siebie nawzajem.

KOPIOWANA MQQSGD_COPY
Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć własną kopię obiektu. Początkowo wszystkie kopie mają takie same atrybuty, ale za pomocą komend MQSC można zmienić każdą kopię, tak aby jej atrybuty różniły się od innych kopii. Atrybuty kopii są ponownie synchronizowane, gdy definicja główna w repozytorium współużytkowanym zostanie zmieniona.

MQQSGD_SHARED
Obiekt ma dyspozycję współużytkowaną. Oznacza to, że we współużytkowanym repozytorium istnieje pojedyncza instancja obiektu, która jest znana wszystkim menedżerom kolejek w grupie współużytkowania kolejek. Gdy menedżer kolejek w grupie uzyskuje dostęp do obiektu, uzyskuje dostęp do pojedynczej współużytkowanej instancji obiektu.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_QSG_DISP z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

QueueAccounting (MQLONG),

Tabela 612. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	

Steruje to gromadzeniem danych rozliczeniowych dla kolejki. Aby dane rozliczeniowe były gromadzone dla tej kolejki, muszą być również włączone dane rozliczeniowe dla tego połączenia przy użyciu atrybutu QMGR ACCTQ lub pola Options w strukturze MQCNO wywołania MQCONNX.

Ten atrybut ma jedną z następujących wartości:

MQMON_Q_MGR
Dane rozliczeniowe dla tej kolejki są gromadzone na podstawie ustawienia atrybutu ACCTQ menedżera kolejek (QMGR). Jest to ustawienie domyślne.

MQMON_WYŁ
Nie kolekcjonuj danych rozliczeniowych dla tej kolejki.

MQMON_ON
Zgromadź dane rozliczeniowe dla tej kolejki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_ACCOUNTING_Q z wywołaniem MQINQ.

QueueMonitoring (MQLONG)
Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kolejek.

Tabela 613. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to jedna z następujących wartości:

MQMON_Q_MGR
Zgromadź dane monitorowania zgodnie z ustawieniem atrybutu menedżera kolejek **QueueMonitoring** . Jest to wartość domyślna.

MQMON_WYŁ
Kolekcjonowanie danych monitorowania bezpośredniego jest wyłączone dla tej kolejki.

MQMON_NISKI
Jeśli wartość atrybutu menedżera kolejek systemu **QueueMonitoring** jest inna niż MQMON_NONE, gromadzenie danych monitorowania bezpośredniego jest włączone, przy niskiej szybkości gromadzenia danych dla tej kolejki.

MQMON_MEDIUM
Jeśli wartość atrybutu menedżera kolejek systemu **QueueMonitoring** jest inna niż MQMON_NONE, gromadzenie danych monitorowania bezpośredniego jest włączone z umiarkowaną szybkością gromadzenia danych dla tej kolejki.

MQMON_HIGH
Jeśli wartość atrybutu menedżera kolejek systemu **QueueMonitoring** jest inna niż MQMON_NONE, gromadzenie danych monitorowania bezpośredniego jest włączone z dużą szybkością gromadzenia danych dla tej kolejki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_MONITORING_Q z wywołaniem MQINQ.

QueueStatistics (MQCHAR12)

Tabela 614. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	

Steruje to gromadzeniem danych statystycznych dla kolejki.

Ten atrybut ma jedną z następujących wartości:

MQMON_Q_MGR
Dane rozliczeniowe dla tej kolejki są gromadzone na podstawie ustawienia atrybutu STATQ menedżera kolejek. Jest to ustawienie domyślne.

MQMON_WYŁ
Wyłącz gromadzenie danych statystycznych dla tej kolejki.

MQMON_ON
Włącz gromadzenie danych statystycznych dla tej kolejki.

Typ kolejki (MQLONG)

Tabela 615. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Jest to typ kolejki; ma jedną z następujących wartości:

ALIAS MQQT
Definicja kolejki aliasowej.

MQQT_CLUSTER
Kolejka klastra.

MQQT_LOCAL,
Kolejka lokalna.

MQQT_REMOTE
Lokalna definicja kolejki zdalnej.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_Q_TYPE z wywołaniem MQINQ.

RemoteQMgrNazwa (MQCHAR48)

Tabela 616. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
			X	

Jest to nazwa zdalnego menedżera kolejek, w którym zdefiniowano kolejkę **RemoteQName** . Jeśli kolejka **RemoteQName** ma wartość **QSGDisp** równą MQQSGD_COPY lub MQQSGD_SHARED, **RemoteQMgrName** może być nazwą grupy współużytkowania kolejek, która jest właścicielem **RemoteQName**.

Jeśli aplikacja otwiera lokalną definicję kolejki zdalnej, pole **RemoteQMgrName** nie może być puste i nie może być nazwą lokalnego menedżera kolejek. Jeśli parametr **XmitQName** jest pusty, jako kolejka transmisji używana jest kolejka lokalna o nazwie takiej samej jak **RemoteQMgrName** . Jeśli nie ma kolejki o nazwie **RemoteQMgrName**, używana jest kolejka identyfikowana przez atrybut **DefXmitQName** menedżera kolejek.

Jeśli ta definicja jest używana dla aliasu menedżera kolejek, **RemoteQMgrName** jest nazwą menedżera kolejek, który jest aliasowany. Może to być nazwa lokalnego menedżera kolejek. W przeciwnym razie, jeśli parametr **XmitQName** jest pusty podczas otwierania, musi istnieć kolejka lokalna o nazwie takiej samej jak **RemoteQMgrName**. Ta kolejka jest używana jako kolejka transmisji.

Jeśli ta definicja jest używana dla aliasu odpowiedzi, ta nazwa jest nazwą menedżera kolejek, który ma być menedżerem kolejek **ReplyToQMgr**.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REMOTE_Q_MGR_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

RemoteQName (MQCHAR48)

Tabela 617. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
			X	

Jest to nazwa kolejki znana w zdalnym menedżerze kolejek *RemoteQMgrName*.

Jeśli aplikacja otwiera lokalną definicję kolejki zdalnej, w momencie otwarcia *RemoteQName* nie może być pusta.

Jeśli ta definicja jest używana dla definicji aliasu menedżera kolejek, po otwarciu *RemoteQName* musi być pusta.

Jeśli definicja jest używana dla aliasu odpowiedzi, ta nazwa jest nazwą kolejki, która ma być nazwą *ReplyToQ*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_REMOTE_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

RetentionInterval (MQLONG)

Jest to okres przechowywania kolejki. Po upływie tego czasu kolejka zostanie zakwalifikowana do usunięcia.

Tabela 618. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Czas jest mierzony w godzinach, licząc od daty i godziny utworzenia kolejki. Data i godzina utworzenia kolejki są rejestrowane w atrybutach **CreationDate** i **CreationTime** .

Te informacje umożliwiają aplikacji konserwacyjnej lub operatorowi identyfikowanie i usuwanie kolejek, które nie są już wymagane.

Uwaga: Menedżer kolejek nigdy nie podejmuje żadnych działań w celu usunięcia kolejek na podstawie tego atrybutu lub w celu zapobieżenia usunięciu kolejek z okresem przechowywania, który nie utracił ważności. Użytkownik jest odpowiedzialny za wykonanie wszystkich wymaganych działań.

Użyj realistycznego przedziału czasu przechowywania, aby zapobiec gromadzeniu trwałych kolejek dynamicznych (patrz atrybut DefinitionType). Atrybut ten może być jednak również używany z predefiniowanymi kolejkami.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_RETENTION_INTERVAL z wywołaniem MQINQ.

Zasięg (MQLONG)

Ta opcja określa, czy pozycja dla tej kolejki istnieje również w katalogu komórki.

Tabela 619. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	

Katalog komórki jest udostępniany przez instalowalną usługę nazw. Jest to jedna z następujących wartości:

MQSCO_Q_MGR
Definicja kolejki ma zasięg menedżera kolejek: definicja kolejki nie wykracza poza menedżera kolejek, który jest jej właścicielem. Aby otworzyć kolejkę dla danych wyjściowych z innego menedżera kolejek, należy określić nazwę menedżera kolejek będącego właścicielem lub inny menedżer kolejek musi mieć lokalną definicję kolejki.

MQSCO_CELL
Definicja kolejki ma zasięg komórki: definicja kolejki jest również umieszczana w katalogu komórki dostępnym dla wszystkich menedżerów kolejek w komórce. Kolejkę można otworzyć w celu wyjścia z dowolnego menedżera kolejek w komórce, określając nazwę kolejki. Nazwa menedżera kolejek, do którego należy kolejka, nie musi być określona. Jednak definicja kolejki nie jest dostępna dla żadnego menedżera kolejek w komórce, który ma również lokalną definicję kolejki o tej nazwie, ponieważ definicja lokalna ma pierwszeństwo.

Katalog komórki jest udostępniany przez instalowalną usługę nazw.

Kolejki modelowe i dynamiczne nie mogą mieć zasięgu komórki.

Ta wartość jest poprawna tylko wtedy, gdy skonfigurowano usługę nazw obsługującą katalog komórki.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SCOPE z wywołaniem MQINQ.

Obsługa tego atrybutu podlega następującym ograniczeniom:

- W systemie IBM iatrybut jest obsługiwany, ale poprawny jest tylko parametr MQSCO_Q_MGR.
- W systemie z/OSatrybut nie jest obsługiwany.

Możliwość współużytkowania (MQLONG)

Wskazuje, czy kolejka może być otwierana na potrzeby wielokrotnego współbieżnego wprowadzania.

Tabela 620. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to jedna z następujących wartości:

MQQA_SHAREABLE
Kolejka jest współużytkowana.

Dozwolonych jest wiele operacji otwierania z opcją MQOO_INPUT_SHARED.

Tabela MQQA_NOT_SHAREABLE
Nie można współużytkować kolejki.

Wywołanie MQOPEN z opcją MQOO_INPUT_SHARED jest traktowane jako wywołanie MQOO_INPUT_EXCLUSIVE.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_SHAREABILITY z wywołaniem MQINQ.

StorageClass (MQCHAR8)

Jest to nazwa zdefiniowana przez użytkownika, która definiuje fizyczną pamięć używaną do przechowywania kolejki. W praktyce komunikat jest zapisywany na dysk tylko wtedy, gdy musi być zrzucany z buforu pamięci.

Tabela 621. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_STORAGE_CLASS z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_STORAGE_CLASS_LENGTH.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

TriggerControl (MQLONG)

Ta opcja określa, czy komunikaty wyzwalacza są zapisywane w kolejce inicjującej w celu uruchomienia aplikacji do obsługi kolejki.

Tabela 622. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to jeden z następujących elementów:

ZMQTC_OFF
Dla tej kolejki nie będą zapisywane żadne komunikaty wyzwalacza. W tym przypadku wartość *TriggerType* jest nieistotna.

MQTC_WŁ
Komunikaty wyzwalacza mają być zapisywane dla tej kolejki, gdy wystąpią odpowiednie zdarzenia wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_CONTROL z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerData (MQCHAR64)

Są to dane w formacie wolnym, które menedżer kolejek wstawia do komunikatu wyzwalacza, gdy komunikat pojawiający się w tej kolejce powoduje zapisanie komunikatu wyzwalacza w kolejce inicjującej.

Tabela 623. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Treść tych danych nie ma znaczenia dla menedżera kolejek. Ma ona znaczenie albo dla aplikacji monitorującej wyzwalacz, która przetwarza kolejkę inicjującą, albo dla aplikacji uruchamianej przez monitor wyzwalacza.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_TRIGGER_DATA z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET. Długość tego atrybutu jest określona przez parametr MQ_TRIGGER_DATA_LENGTH.

TriggerDepth (MQLONG)

Tabela 624. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to liczba komunikatów o priorytecie *TriggerMsgPriority* lub większym, które muszą znajdować się w kolejce przed zapisaniem komunikatu wyzwalacza. Ma to zastosowanie, gdy parametr *TriggerType* jest ustawiony na wartość MQTT_DEPTH. Wartość parametru *TriggerDepth* wynosi jeden lub więcej. W przeciwnym razie ten atrybut nie jest używany.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_DEPTH z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerMsgPriorytet (MQLONG)

Jest to priorytet komunikatu, poniżej którego komunikaty nie przyczyniają się do generowania komunikatów wyzwalacza (oznacza to, że menedżer kolejek ignoruje te komunikaty podczas podejmowania decyzji o generowaniu komunikatu wyzwalacza).

Tabela 625. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Wartość *TriggerMsgPriority* może być z zakresu od 0 (najniższy priorytet) do *MaxPriority* (najwyższy priorytet; patrz atrybut MaxPriority). Wartość zero powoduje, że wszystkie komunikaty przyczyniają się do generowania komunikatów wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_MSG_PRIORITY z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

TriggerType (MQLONG)

Steruje to warunkami, w jakich komunikaty wyzwalacza są zapisywane w wyniku komunikatów przychodzących do tej kolejki.

Tabela 626. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Ma jedną z następujących wartości:

MQTT_BRAK
W wyniku komunikatów w tej kolejce nie są zapisywane żadne komunikaty wyzwalacza. Ma to taki sam efekt, jak ustawienie parametru *TriggerControl* na wartość MQTC_OFF.

MQTT_FIRST
Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce zmieni się z 0 na 1.

MQTT_EVERY
Komunikat wyzwalacza jest zapisywany za każdym razem, gdy w kolejce pojawi się komunikat o priorytecie *TriggerMsgPriority* lub większym.

MQTT_DEPTH
Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce jest równa lub większa niż *TriggerDepth*. Po zapisaniu komunikatu wyzwalacza *TriggerControl* jest ustawiany na wartość MQTC_OFF, aby zapobiec dalszemu wyzwalaniu, dopóki nie zostanie jawnie ponownie włączony.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_TRIGGER_TYPE z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

Użycie (MQLONG)
Wskazuje, do czego jest używana kolejka.

Tabela 627. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to jedna z następujących wartości:

MQUS_NORMAL
Jest to kolejka używana przez aplikacje podczas umieszczania i pobierania komunikatów. Kolejka nie jest kolejką transmisji.

MQUS_TRANSMISSION.
Jest to kolejka używana do przechowywania komunikatów przeznaczonych dla zdalnych menedżerów kolejek. Gdy aplikacja wysyła komunikat do kolejki zdalnej, menedżer kolejek lokalnych zapisuje komunikat tymczasowo w odpowiedniej kolejce transmisji w specjalnym formacie. Następnie agent kanału komunikatów odczytuje komunikat z kolejki transmisji i transportuje go do zdalnego menedżera kolejek. Więcej informacji na temat konfigurowania zdalnego administrowania zawiera sekcja Konfigurowanie zdalnego administrowania menedżerami kolejek.

Tylko aplikacje uprzywilejowane mogą otwierać kolejkę transmisji dla MQOO_OUTPUT w celu bezpośredniego umieszczania w niej komunikatów. Zwykle robią to tylko aplikacje narzędziowe. Upewnij się, że format danych komunikatu jest poprawny (patrz sekcja “MQXQH-nagłówek kolejki transmisji” na stronie 637). lub mogą wystąpić błędy podczas procesu transmisji. Kontekst nie jest przekazywany ani ustawiany, chyba że określono jedną z opcji kontekstu MQPMO_ * _CONTEXT.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_USAGE z wywołaniem MQINQ.

XmitQName (MQCHAR48)
Jest to nazwa kolejki transmisji. Jeśli ten atrybut nie jest pusty podczas otwierania, dla kolejki zdalnej lub dla definicji aliasu menedżera kolejek, określa nazwę lokalnej kolejki transmisji, która ma być używana do przekazywania komunikatu.

Tabela 628. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
			X	

Jeśli parametr **XmitQName** jest pusty, jako kolejka transmisji używana jest kolejka lokalna o nazwie takiej samej jak **RemoteQMgrName** . Jeśli nie ma kolejki o nazwie **RemoteQMgrName**, używana jest kolejka identyfikowana przez atrybut **DefXmitQName** menedżera kolejek.

Ten atrybut jest ignorowany, jeśli definicja jest używana jako alias menedżera kolejek, a **RemoteQMgrName** jest nazwą lokalnego menedżera kolejek. Atrybut nie jest również brany pod uwagę, jeśli definicja jest używana jako definicja aliasu kolejki zwrotnej.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_XMIT_Q_NAME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_Q_NAME_LENGTH.

Atrybuty listy nazw

W poniższej tabeli znajduje się podsumowanie atrybutów specyficznych dla list nazw. Atrybuty są opisane w porządku alfabetycznym.

Listy nazw są obsługiwane we wszystkich systemach IBM MQ oraz w systemach IBM MQ MQI clients połączonych z tymi systemami.

Uwaga: Nazwy atrybutów przedstawione w tej sekcji są opisowymi nazwami używanymi w wywołaniach MQINQ i MQSET. Nazwy są takie same, jak w przypadku komend PCF. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja Komendy MQSC .

Tabela 629. Atrybuty listy nazw

Atrybut	Opis
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
NameCount	Liczba nazw na liście nazw
NamelistDesc	Opis listy nazw
NamelistName	Nazwa listy nazw
Nazwy	Lista nazw *NameCount*
NamelistType	Typ listy nazw
QSGDISP	Dyspozycja grupy współużytkowania kolejki

AlterationDate (MQCHAR12)

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_TIME_LENGTH.

NameCount (MQLONG)

Liczba nazw na liście nazw. Jest ona większa lub równa zero. Zdefiniowana jest następująca wartość:

MQNC_MAX_NAMELIST_NAME_COUNT,
Maksymalna liczba nazw na liście nazw.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_NAME_COUNT z wywołaniem MQINQ.

NamelistDesc (MQCHAR64)

To pole służy do opisu komentarza; jego wartość jest określana przez proces definiowania. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole to może zawierać znaki DBCS (maksymalna długość pola wynosi 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_NAMELIST_DESC z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez wartość MQ_NAMELIST_DESC_LENGTH.

NamelistName (MQCHAR48)

Jest to nazwa listy nazw, która jest zdefiniowana w menedżerze kolejek lokalnych. Więcej informacji na temat nazw list nazw zawiera sekcja Nazwy innych obiektów .

Każda lista nazw ma nazwę inną niż nazwy innych list nazw należących do menedżera kolejek, ale może duplikować nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_NAMELIST_NAME z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez wartość MQ_NAMELIST_NAME_LENGTH.

NamelistType (MQLONG)

Określa rodzaj nazw na liście nazw i wskazuje sposób użycia listy nazw. Jest to jedna z następujących wartości:

MQNT_NONE
Lista nazw bez przypisanego typu.

MQNT_Q,
Lista nazw zawierająca nazwy kolejek.

MQNT_CLUSTER
Lista nazw zawierająca nazwy klastrów.

MQNT_AUTH_INFO,
Lista nazw zawierająca nazwy obiektów informacji uwierzytelniającej.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_NAMELIST_TYPE z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

Nazwy (MQCHAR48xNameCount)

Jest to lista nazw *NameCount* , gdzie każda nazwa jest nazwą obiektu zdefiniowanego w menedżerze kolejek lokalnych. Więcej informacji na temat nazw obiektów zawiera sekcja Reguły nazewnictwa obiektów IBM MQ.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_NAMES z wywołaniem MQINQ.

Długość każdej nazwy na liście jest określona przez wartość MQ_OBJECT_NAME_LENGTH.

QSGDisp (MQLONG)

Określa dyspozycję listy nazw. Jest to jedna z następujących wartości:

MQQSGD_Q_MGR
Obiekt ma dyspozycję menedżera kolejek: definicja obiektu jest znana tylko menedżerowi kolejek lokalnych; definicja nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć obiekt o takiej samej nazwie i typie jak bieżący obiekt, ale są to oddzielne obiekty i nie ma między nimi korelacji. Ich atrybuty nie są ograniczone do siebie nawzajem.

KOPIOWANA MQQSGD_COPY
Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć własną kopię obiektu. Początkowo wszystkie kopie mają takie same atrybuty, ale można zmienić każdą kopię, używając komend MQSC, tak aby jej atrybuty różniły się od atrybutów innych kopii. Atrybuty kopii są ponownie synchronizowane, gdy definicja główna w repozytorium współużytkowanym zostanie zmieniona.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_QSG_DISP z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

Atrybuty definicji procesów

W poniższej tabeli znajduje się podsumowanie atrybutów specyficznych dla definicji procesów. Atrybuty są opisane w porządku alfabetycznym.

Uwaga: Nazwy atrybutów w tej sekcji są opisowymi nazwami używanymi w wywołaniach MQINQ i MQSET; nazwy są takie same, jak w przypadku komend PCF. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja Komendy MQSC .

Tabela 630. Atrybuty definicji procesów

Atrybut	Opis
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
ApplId	Identyfikator aplikacji
ApplType	Typ aplikacji
EnvData	Dane środowiska
ProcessDesc	Opis procesu
ProcessName	Nazwa procesu
QSGDISP	Dyspozycja grupy współużytkowania kolejki
UserData	Dane użytkownika

AlterationDate (MQCHAR12)

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_DATE z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_DATE_LENGTH.

AlterationTime (MQCHAR8)

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ALTERATION_TIME z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_TIME_LENGTH.

ApplId (MQCHAR256)

Jest to łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona. Te informacje są używane przez aplikację monitorującą wyzwalacz, która przetwarza komunikaty w kolejce inicjującej. Informacje te są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalającego.

Znaczenie parametru *ApplId* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniony przez IBM MQ wymaga, aby *ApplId* była nazwą programu wykonywalnego. Poniższe uwagi dotyczą wskazanych środowisk:

- W systemie z/OS *ApplId* musi być:
  - Identyfikator transakcji CICS dla aplikacji uruchomionych za pomocą transakcji CKTI monitora wyzwalacza CICS
  - Identyfikator transakcji IMS dla aplikacji uruchomionych za pomocą monitora wyzwalacza IMS CSQQTRMN
- W systemie Windowsnazwa programu może być poprzedzona ścieżką do napędu i katalogu.
- W systemie AIX and Linuxnazwa programu może być poprzedzona ścieżką do katalogu.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_APPL_ID z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_PROCESS_APPL_ID_LENGTH.

ApplType (MQLONG)

Identyfikuje on rodzaj programu, który ma zostać uruchomiony w odpowiedzi na odebranie komunikatu wyzwalacza. Te informacje są używane przez aplikację monitorującą wyzwalacz, która przetwarza komunikaty w kolejce inicjującej. Informacje te są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalającego.

Parametr *ApplType* może mieć dowolną wartość, ale w przypadku typów standardowych zalecane są następujące wartości. Należy ograniczyć typy aplikacji zdefiniowane przez użytkownika do wartości z zakresu od MQAT_USER_FIRST do MQAT_USER_LAST:

MQAT_AIX
Aplikacja AIX (taka sama jak MQAT_UNIX).

MQAT_BATCH
aplikacja wsadowa

MQAT_CICS
CICS .

MQAT_IMS
Aplikacja IMS .

MQAT_IMS_BRIDGE,
Aplikacja mostu IMS .

MQAT_JAVA
Aplikacja Java .

MQAT_MVS
Aplikacja MVS lub TSO (taka sama wartość jak MQAT_ZOS).

MQAT_OS390
Aplikacja OS/390 (taka sama jak MQAT_ZOS).

MQAT_OS400
Aplikacja IBM i .

MQAT_UNIX
Aplikacja UNIX .

MQAT_UNKNOWN
Zastosowanie nieznanego typu.

UŻYTKOWNIK_MQAT
Aplikacja użytkownika.

MQAT_WINDOWS
64-bitowa aplikacja Windows .

MQAT_WINDOWS_NT
32-bitowa aplikacja Windows .

MQAT_WLM
Aplikacja menedżera obciążenia z/OS .

MQAT_ZOS
Aplikacja z/OS .

MQAT_USER_FIRST
Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Tabela MQAT_USER_LAST
Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_APPL_TYPE z wywołaniem MQINQ.

EnvData (MQCHAR128)

Jest to łańcuch znaków zawierający informacje związane ze środowiskiem dotyczące uruchamianej aplikacji. Te informacje są używane przez aplikację monitorującą wyzwalacz, która przetwarza komunikaty w kolejce inicjującej. Informacje te są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalającego.

Znaczenie parametru *EnvData* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniany przez IBM MQ dopisuje *EnvData* do listy parametrów przekazanej do uruchomionej aplikacji. Lista parametrów składa się ze struktury MQTMC2 , po której następuje jedno puste miejsce, po którym następuje łańcuch *EnvData* z usuniętymi odstępami końcowymi. Poniższe uwagi dotyczą wskazanych środowisk:

- W systemie z/OS:
  - Produkt *EnvData* nie jest używany przez aplikacje monitora wyzwalacza udostępniane przez produkt IBM MQ.
  - Jeśli ApplType ma wartość MQAT_WLM, można podać wartości domyślne w polu EnvData w polach ServiceName i ServiceStep w nagłówku informacji o pracy (MQWIH).
- W systemie AIX and Linuxparametr *EnvData* można ustawić na znak & , aby uruchomić uruchomionej aplikacji w tle.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_ENV_DATA z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_PROCESS_ENV_DATA_LENGTH.

ProcessDesc (MQCHAR64)

To pole służy do opisowego komentarza. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole może zawierać znaki DBCS (z maksymalną długością pola wynoszącą 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_PROCESS_DESC z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez wartość MQ_PROCESS_DESC_LENGTH.

ProcessName (MQCHAR48)

Jest to nazwa definicji procesu, która jest zdefiniowana w menedżerze kolejek lokalnych.

Każda definicja procesu ma nazwę inną niż nazwy innych definicji procesów należących do menedżera kolejek. Jednak nazwa definicji procesu może być taka sama, jak nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_PROCESS_NAME z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez wartość MQ_PROCESS_NAME_LENGTH.

QSGDisp (MQLONG)

Określa dyspozycję definicji procesu. Jest to jedna z następujących wartości:

MQQSGD_Q_MGR
Obiekt ma dyspozycję menedżera kolejek: definicja obiektu jest znana tylko menedżerowi kolejek lokalnych; definicja nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć obiekt o takiej samej nazwie i typie jak bieżący obiekt, ale są to oddzielne obiekty i nie ma między nimi korelacji. Ich atrybuty nie są ograniczone do siebie nawzajem.

KOPIOWANA MQQSGD_COPY
Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć własną kopię obiektu. Początkowo wszystkie kopie mają takie same atrybuty, ale można zmienić każdą kopię, używając komend MQSC, tak aby jej atrybuty różniły się od atrybutów innych kopii. Atrybuty kopii są ponownie synchronizowane, gdy definicja główna w repozytorium współużytkowanym zostanie zmieniona.

Aby określić wartość tego atrybutu, należy użyć selektora MQIA_QSG_DISP z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

UserData (MQCHAR128)

`UserData` to łańcuch znaków zawierający informacje o użytkowniku dotyczące aplikacji, która ma zostać uruchomiona. Te informacje są przeznaczone dla aplikacji monitora wyzwalacza, która przetwarza komunikaty w kolejce inicjującej lub aplikacji uruchomionej przez monitor wyzwalacza. Informacje są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalacza.

Znaczenie parametru *UserData* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniony przez IBM MQ przekazuje *UserData* do uruchomionej aplikacji jako część listy parametrów. Lista parametrów składa się ze struktury MQTMC2 (zawierającej łańcuch *UserData*), po której następuje jedno puste miejsce, po którym następuje łańcuch *EnvData* z usuniętymi odstępami końcowymi.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami. W przypadku systemu Microsoft Windowsłańcuch znaków nie może zawierać podwójnych cudzysłowów, jeśli definicja procesu ma zostać przekazana do programu **runmqtrm**.

Aby określić wartość tego atrybutu, należy użyć selektora MQCA_USER_DATA z wywołaniem MQINQ. Długość tego atrybutu jest określona przez wartość MQ_PROCESS_USER_DATA_LENGTH.

Kody powrotu

Dla każdego wywołania interfejsu IBM MQ Message Queue Interface (MQI) i interfejsu IBM MQ Administration Interface (MQAI) kod **zakończenia** i kod **przyczyny** są zwracane przez menedżer kolejek lub przez procedurę wyjścia w celu wskazania powodzenia lub niepowodzenia wywołania.

Wnioski nie mogą zależeć od błędów sprawdzanych w określonej kolejności, z wyjątkiem przypadków, w których zaznaczono inaczej. Jeśli z wywołania może wynikać więcej niż jeden kod zakończenia lub kod przyczyny, zgłoszony błąd zależy od implementacji.

Aplikacje sprawdzający pomyślne zakończenie po wywołaniu funkcji API IBM MQ muszą zawsze sprawdzać kod zakończenia. Nie należy przyjmować wartości kodu zakończenia na podstawie wartości kodu przyczyny.

Kody zakończenia

Parametr kodu zakończenia (*CompCode*) umożliwia programowi wywołującemu szybkie sprawdzenie, czy wywołanie zakończyło się pomyślnie, zostało zakończone częściowo, czy nie.Poniżej znajduje się lista kodów zakończenia z większą ilością szczegółów niż podano w opisach wywołania:

MQCC_OK
Wywołanie zostało zakończone w pełni; wszystkie parametry wyjściowe zostały ustawione. W tym przypadku parametr **Reason** zawsze ma wartość MQRC_NONE.

Ostrzeżenie MQCC
Wywołanie zostało zakończone częściowo. Niektóre parametry wyjściowe mogły zostać ustawione oprócz parametrów wyjściowych *CompCode* i *Reason* . Parametr **Reason** udostępnia dodatkowe informacje o częściowym zakończeniu.

MQCC_FAILED (niepowodzenie MQC)
Przetwarzanie wywołania nie zostało zakończone. Stan menedżera kolejek pozostaje niezmieniony, chyba że zaznaczono inaczej. Parametry wyjściowe *CompCode* i *Reason* zostały ustawione; inne parametry pozostają niezmienione, chyba że zaznaczono inaczej.

Przyczyną może być błąd w aplikacji lub sytuacja zewnętrzna programu, na przykład unieważnienie uprawnień użytkownika. Parametr **Reason** udostępnia dodatkowe informacje o błędzie.

Kody przyczyny

Parametr kodu przyczyny (*Reason*) kwalifikuje parametr kodu zakończenia (*CompCode*).

Jeśli nie ma specjalnego powodu do raportowania, zwracana jest wartość MQRC_NONE. Pomyślne wywołanie zwraca wywołania MQCC_OK i MQRC_NONE.

Jeśli kodem zakończenia jest MQCC_WARNING lub MQCC_FAILED, menedżer kolejek zawsze zgłasza kwalifikującą się przyczynę; szczegóły są podane w każdym opisie wywołania.

Tam, gdzie procedury użytkownika ustawiają kody zakończenia i przyczyny, muszą one być zgodne z tymi regułami. Ponadto wszystkie wartości przyczyn specjalnych zdefiniowane przez procedury zewnętrzne muszą być mniejsze od zera, aby zapewnić, że nie będą kolidować z wartościami zdefiniowanymi przez menedżer kolejek. W razie potrzeby wyjścia mogą ustawiać przyczyny, które zostały już zdefiniowane przez menedżer kolejek.

Kody przyczyny występują również w:

- Pole *Reason* struktury MQDLH
- Pole *Feedback* struktury MQMD

Pełny opis kodów przyczyny można znaleźć w sekcji Komunikaty i kody przyczyny.

Reguły sprawdzania poprawności opcji MQI

Ta sekcja zawiera listę sytuacji, które generują kod przyczyny MQRC_OPTIONS_ERROR z wywołania MQOPEN, MQPUT, MQPUT1, MQGET, MQCLOSE lub MQSUB.

Wywołanie MQOPEN

Opcje wywołania MQOPEN:

- Należy podać co najmniej *jeden* z następujących parametrów:
  - 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]
- Dozwolony jest tylko *jeden* z następujących elementów:
  - MQOO_READ_AHEAD
  - MQOO_NO_READ_AHEAD
  - MQOO_READ_AHEAD_AS_Q_DEF

1. Dozwolony jest tylko *jeden* z następujących elementów:
   - MQOO_INPUT_EXCLUSIVE (wyłączna)
   - MQOO_INPUT_SHARED
   - MQOO_INPUT_AS_Q_DEF
2. Dozwolony jest tylko *jeden* z następujących elementów:
   - MQOO_BIND_ON_OPEN-OTWARTE
   - MQOO_BIND_NOT_FIXED (NIEUSTALONY)
   - MQOO_BIND_ON_GROUP
   - MQOO_BIND_AS_Q_DEF

   Uwaga: Wymienione wcześniej opcje wykluczają się wzajemnie. Ponieważ jednak wartość opcji MQOO_BIND_AS_Q_DEF wynosi zero, podanie jej z jedną z dwóch pozostałych opcji wiązania nie powoduje wystąpienia kodu przyczyny MQRC_OPTIONS_ERROR. Komenda MQOO_BIND_AS_Q_DEF jest udostępniana w celu wspomagania dokumentacji programu.

- Jeśli podano opcję MQOO_SAVE_ALL_CONTEXT, należy również podać jedną z opcji MQOO_INPUT_ *.
- Jeśli określono jedną z opcji MQOO_SET_ * _CONTEXT lub MQOO_PASS_ * _CONTEXT, należy również określić opcję MQOO_OUTPUT.
- Jeśli określono opcję MQOO_CO_OP, należy również określić opcję MQOO_BROWSE.
- Jeśli określono opcję MQOO_NO_MULTICAST, należy również określić opcję MQOO_OUTPUT.

Wywołanie MQPUT

Dla opcji put-message:

- Kombinacja MQPMO_SYNCPOINT i MQPMO_NO_SYNCPOINT nie jest dozwolona.
- Dozwolony jest tylko *jeden* z następujących elementów:
  - MQPMO_DEFAULT_CONTEXT,
  - MQPMO_NO_CONTEXT
  - MQPMO_PASS_ALL_CONTEXT
  - MQPMO_PASS_IDENTITY_CONTEXT,
  - MQPMO_SET_ALL_CONTEXT
  - MQPMO_SET_IDENTITY_CONTEXT,
- Dozwolony jest tylko *jeden* z następujących elementów:

  - MQPMO_ASYNC_RESPONSE
  - MQPMO_SYNC_RESPONSE
  - MQPMO_ODPOWIEDŹ_AS_TOPIC_DEF
  - MQPMO_ODPOWIEDŹ_AS_Q_DEF
- Uprawnienie MQPMO_ALTERNATE_USER_AUTHORITY jest niedozwolone (jest poprawne tylko w przypadku wywołania MQPUT1).

Wywołanie MQPUT1

W przypadku opcji put-message reguły są takie same jak w przypadku wywołania MQPUT, z wyjątkiem następujących:

- Parametr MQPMO_ALTERNATE_USER_AUTHORITY jest dozwolony.
- Parametr QPMO_LOGICAL_ORDER jest niedozwolony.

Wywołanie MQGET

W przypadku opcji get-message:

- Dozwolony jest tylko *jeden* z następujących elementów:
  - MQGMO_NO_SYNCPOINT
  - MQGMO_SYNCPOINT
  - MQGMO_SYNCPOINT_IF_PERSISTENT,
- Dozwolony jest tylko *jeden* z następujących elementów:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_MSG_UNDER_CURSOR
- Parametr MQGMO_SYNCPOINT nie jest dozwolony z żadną z następujących wartości:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - BLOKADA MQGMO_LOCK
  - MQGMO_UNLOCK (odblokowanie MQGMO)
- Parametr MQGMO_SYNCPOINT_IF_PERSISTENT nie jest dozwolony z żadną z następujących wartości:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
  - MQGMO_COMPLETE_MSG
  - MQGMO_UNLOCK (odblokowanie MQGMO)
- Opcja MQGMO_MARK_SKIP_BACKOUT wymaga określenia opcji MQGMO_SYNCPOINT.
- Kombinacja komend MQGMO_WAIT i MQGMO_SET_SIGNAL nie jest dozwolona.
- Jeśli określono parametr MQGMO_LOCK, należy również określić jedną z następujących wartości:
  - MQGMO_BROWSE_FIRST
  - MQGMO_BROWSE_MSG_UNDER_CURSOR
  - MQGMO_BROWSE_NEXT
- Jeśli określono parametr MQGMO_UNLOCK, dozwolone są tylko następujące wartości:

- MQGMO_NO_SYNCPOINT
- MQGMO_NO_WAIT

Wywołanie MQCLOSE

W przypadku opcji wywołania MQCLOSE:

- Kombinacja MQCO_DELETE i MQCO_DELETE_PURGE nie jest dozwolona.
- Dozwolony jest tylko jeden z następujących elementów:
  - MQCO_KEEP_SUB
  - MQCO_REMOVE_SUB

Wywołanie MQSUB

W przypadku opcji wywołania MQSUB:

- Należy określić co najmniej jeden z następujących elementów:
  - MQSO_ALTER (zmiana MQSO)
  - MQSO_RESUME
  - MQSO_CREATE
- Dozwolony jest tylko jeden z następujących elementów:
  - MQSO_DURABLE
  - MQSO_NON_DURABLE

  Uwaga: Wymienione wcześniej opcje wykluczają się wzajemnie. Ponieważ jednak wartość parametru MQSO_NON_DURABLE wynosi zero, podanie go z parametrem MQSO_DURABLE nie powoduje wystąpienia kodu przyczyny MQRC_OPTIONS_ERROR. Komenda MQSO_NON_DURABLE jest udostępniana w celu wspomagania dokumentacji programu.
- Kombinacja opcji MQSO_GROUP_SUB i MQSO_MANAGED nie jest dozwolona.
- Opcja MQSO_GROUP_SUB wymaga określenia identyfikatora MQSO_SET_CORREL_ID.
- Dozwolony jest tylko jeden z następujących elementów:
  - MQSO_ANY_USERID
  - MQSO_FIXED_USERID (Identyfikator stałego użytkownika)
- Parametr MQSO_NEW_PUBLIC TYLKO jest dozwolony w połączeniu z:
  - MQSO_CREATE
  - MQSO_ALTER, jeśli w oryginalnej subskrypcji ustawiono parametr MQSO_NEW_PUBLIC
- Kombinacja wartości MQSO_PUBLIC ATIONS_ON_REQUEST i SubLevel większej niż 1 nie jest dozwolona.
- Dozwolony jest tylko jeden z następujących elementów:
  - MQSO_WILDCARD_CHAR
  - MQSO_WILDCARD_TOPIC,
- Opcja MQSO_NO_MULTICAST wymaga określenia opcji MQSO_MANAGED.

Komunikaty komend publikowania/subskrypcji w kolejce

Aplikacja może używać komunikatów komend MQRFH2 do sterowania umieszczoną w kolejce aplikacją publikującą/subskrybującą.

Aplikacja, która używa produktu MQRFH2 do publikowania/subskrybowania, może wysyłać następujące komunikaty komend do systemu SYSTEM.BROKER.CONTROL.QUEUE:

- “Usuń komunikat publikacji” na stronie 908
- “Komunikat wyrejestrowywania subskrybenta” na stronie 909
- “Publikuj komunikat” na stronie 913
- “Komunikat rejestrowania subskrybenta” na stronie 916
- “Komunikat żądania aktualizacji” na stronie 920

Jeśli są zapisywane kolejkowane aplikacje publikowania/subskrypcji, należy zrozumieć te komunikaty, komunikat odpowiedzi menedżera kolejek i deskryptor komunikatu (MQMD). Należy zapoznać się z następującymi informacjami:

- “Komunikat odpowiedzi menedżera kolejek” na stronie 923
- “Ustawienia MQMD dla publikacji przekazywanych przez menedżer kolejek” na stronie 928
- “Ustawienia MQMD w komunikatach odpowiedzi menedżera kolejek” na stronie 929
- “Kody przyczyny publikowania/subskrypcji” na stronie 924

Komendy znajdują się w folderze psc w polu **NameValueData** nagłówka MQRFH2 . Komunikat, który może zostać wysłany przez broker w odpowiedzi na komunikat komendy, znajduje się w folderze pscr .

Opisy poszczególnych komend zawierają listę właściwości, które mogą znajdować się w folderze. Jeśli nie określono inaczej, właściwości są opcjonalne i mogą występować tylko raz.

Nazwy właściwości są wyświetlane jako <Command>.

Wartości muszą być w formacie łańcucha, na przykład: Publish.

Stała łańcuchowa reprezentująca wartość właściwości jest wyświetlana w nawiasach, na przykład: (MQPSC_PUBLISH).

Stałe łańcuchowe są definiowane w pliku nagłówkowym cmqpsc.h , który jest dostarczany z menedżerem kolejek.

Usuń komunikat publikacji

Komunikat komendy **Delete Publication** jest wysyłany do menedżera kolejek z publikatora lub z innego menedżera kolejek w celu poinformowania menedżera kolejek o usunięciu wszystkich zachowanych publikacji dla określonych tematów.

Ten komunikat jest wysyłany do kolejki monitorowanej przez umieszczony w kolejce interfejs publikowania/subskrypcji menedżera kolejek.

Kolejka wejściowa powinna być kolejką, do której została wysłana oryginalna publikacja.

Jeśli użytkownik ma uprawnienia do niektórych, ale nie do wszystkich, tematów określonych w komunikacie komendy **Delete Publication** , usuwane są tylko te tematy. Komunikat **Broker Response** wskazuje, które tematy nie zostały usunięte.

Podobnie, jeśli komenda **Publish** zawiera więcej niż jeden temat, komenda **Delete Publication** zgodna z niektórymi, ale nie wszystkimi, z tych tematów usunie tylko publikacje z tematów określonych w komendzie **Delete Publication** .

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD) wymaganych podczas wysyłania komunikatu komendy do menedżera kolejek zawiera sekcja “Ustawienia MQMD dla publikacji przekazywanych przez menedżer kolejek” na stronie 928 .

Właściwości

Komenda (*MQPSC_COMMAND*)

Wartością jest DeletePub (*MQPSC_DELETE_PUBLICATION*).

Ta właściwość musi być określona.

Temat > (*MQPSC_TOPIC*)
Wartością jest łańcuch zawierający temat, dla którego mają zostać usunięte zachowane publikacje. W łańcuchu można umieścić znaki wieloznaczne, aby usunąć publikacje z więcej niż jednego tematu.

Ta właściwość musi być określona; może być powtarzana dla dowolnej liczby tematów.

DelOpt (*MQPSC_DELETE_OPTION*)
Właściwość opcji usuwania może przyjmować jedną z następujących wartości:

`Lokalna` (*MQPSC_LOCAL*)

Wszystkie zachowane publikacje dla określonych tematów są usuwane w menedżerze kolejek lokalnych (czyli w menedżerze kolejek, do którego wysyłany jest ten komunikat), niezależnie od tego, czy zostały opublikowane przy użyciu opcji `Lokalne` .

Nie ma to wpływu na publikacje w innych menedżerach kolejek.

`Brak` (*MQPSC_NONE*)

Wszystkie opcje przyjmują wartości domyślne. Efekt jest taki sam, jak pominięcie właściwości `DelOpt` . Jeśli w tym samym czasie zostaną podane inne opcje, opcja `Brak` zostanie zignorowana.

Domyślnie, jeśli ta właściwość jest pominięta, wszystkie zachowane publikacje dla określonych tematów są usuwane we wszystkich menedżerach kolejek w sieci, niezależnie od tego, czy zostały opublikowane za pomocą opcji `Lokalne` .

Przykład

Poniżej przedstawiono przykład danych `NameValue` dla komunikatu komendy **`Delete Publication`** . Jest ona używana przez przykładową aplikację do usuwania w lokalnym menedżerze kolejek zachowanej publikacji, która zawiera najnowszy wynik dopasowania między Team1 i Team2.

```
<psc>
  <Command>DeletePub</Command>
  <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
  <DelOpt>Local</DelOpt>
</psc>
```

Komunikat wyrejestrowywania subskrybenta

Komunikat komendy **`Deregister Subscriber`** jest wysyłany do menedżera kolejek przez subskrybenta lub przez inną aplikację w imieniu subskrybenta, aby wskazać, że nie chce on już odbierać komunikatów zgodnych z podanymi parametrami.

Ten komunikat jest wysyłany do systemu SYSTEM.BROKER.CONTROL.QUEUE-kolejka sterująca menedżera kolejek. Użytkownik musi mieć uprawnienia niezbędne do umieszczenia komunikatu w tej kolejce.

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD), które są wymagane podczas wysyłania komunikatu komendy do menedżera kolejek, zawiera sekcja Ustawienia deskryptora MQMD dla publikacji przekazanych przez menedżer kolejek .

Pojedynczą subskrypcję można wyrejestrowywać, określając odpowiedni temat, punkt subskrypcji i wartości filtru oryginalnej subskrypcji. Jeśli którakolwiek z tych wartości nie została określona (czyli przyjmowała wartości domyślne) w oryginalnej subskrypcji, powinna zostać pominięta podczas wyrejestrowyowania subskrypcji.

Wszystkie subskrypcje subskrybenta lub grupy subskrybentów można wyrejestrowywać za pomocą opcji `DeregAll` . Jeśli na przykład zostanie określona opcja `DeregAll` wraz z punktem subskrypcji (ale bez tematu lub filtru), wszystkie subskrypcje dla subskrybenta w określonym punkcie subskrypcji zostaną wyrejestrowane niezależnie od tematu i filtru. Dozwolona jest dowolna kombinacja tematu, filtru i punktu subskrypcji. Jeśli wszystkie trzy zostały określone, może być zgodna tylko jedna subskrypcja, a opcja `DeregAll` jest ignorowana.

Komunikat musi zostać wysłany przez subskrybenta, który zarejestrował subskrypcję. Potwierdza to sprawdzenie identyfikatora użytkownika subskrybenta.

Subskrypcje mogą być również wyrejestrowywane przez administratora systemu za pomocą komend MQSC lub PCF. Jednak subskrypcje zarejestrowane w tymczasowej kolejce dynamicznej są powiązane z kolejką, a nie tylko z nazwą kolejki. Jeśli kolejka zostanie usunięta jawnie lub przez aplikację rozłączającą się z menedżerem kolejek, nie będzie już możliwe użycie komendy **Deregister Subscriber** w celu wyrejestrowania subskrypcji dla tej kolejki. Subskrypcje można wyrejestrowywać za pomocą środowiska roboczego programisty i są one automatycznie usuwane przez menedżer kolejek przy następnej operacji dopasowania publikacji do subskrypcji lub przy następnym restarcie menedżera kolejek. W normalnych okolicznościach aplikacje powinny wyrejestrować swoje subskrypcje przed usunięciem kolejki lub odłączeniem się od menedżera kolejek.

Jeśli subskrybent wysyła komunikat w celu wyrejestrowania subskrypcji i odbiera komunikat odpowiedzi informujący o pomyślnym przetworzeniu tej subskrypcji, niektóre publikacje mogą nadal docierać do kolejki subskrybenta, jeśli były przetwarzane przez menedżer kolejek w tym samym czasie, w którym subskrypcja była wyrejestrowana. Jeśli komunikaty nie są usuwane z kolejki, w kolejce subskrybenta może występować narastanie nieprzetworzonych komunikatów. Jeśli aplikacja wykona pętlę zawierającą wywołanie MQGET z odpowiednim identyfikatorem `CorrelId` po okresie uśpienia przez chwilę, komunikaty te zostaną usunięte z kolejki.

Podobnie, jeśli subskrybent używa trwałej kolejki dynamicznej, a następnie wyrejestrowuje i zamyka kolejkę za pomocą opcji *MQCO_DELETE_PURGE* w wywołaniu MQCLOSE, kolejka może nie być pusta. Jeśli jakiekolwiek publikacje z menedżera kolejek nie zostały jeszcze zatwierdzone podczas usuwania kolejki, wywołanie MQCLOSE wygeneruje kod powrotu MQRC_Q_NOT_EMPTY. Aplikacja może uniknąć tego problemu przez uśpienie i ponowne wywołanie wywołania MQCLOSE od czasu do czasu.

Właściwości

Komenda (*MQPSC_COMMAND*)
Wartością jest `DeregSub` (*MQPSC_DEREGISTER_SUBSCRIBER*).

Ta właściwość musi być określona.

Temat (*MQPSC_TOPIC*)
Wartością jest łańcuch zawierający temat, który ma zostać wyrejestrowany.

Tę właściwość można opcjonalnie powtórzyć, jeśli ma zostać wyrejestrowanych wiele tematów. Można ją pominąć, jeśli opcja `DeregAll` jest określona w `<RegOpt>`.

Określone tematy mogą być podzbiorem tych, które są zarejestrowane, jeśli subskrybent chce zachować subskrypcje dla innych tematów. Znaki wieloznaczne są dozwolone, ale łańcuch tematu zawierający znaki wieloznaczne musi być dokładnie zgodny z odpowiednim łańcuchem określonym w komunikacie komendy **Deregister Subscriber** .

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Wartość jest łańcuchem określającym punkt subskrypcji, od którego subskrypcja ma zostać odłączona.

Ta właściwość nie może się powtarzać. Można ją pominąć, jeśli określono `< Topic>` lub jeśli w parametrze `<RegOpt>`określono wartość `DeregAll` . Jeśli ta właściwość zostanie pominięta, wykonywane są następujące działania:

- Jeśli **nie** określono opcji `DeregAll`, subskrypcje zgodne z właściwością `< Topic>` (oraz z właściwością `< Filter >` , jeśli istnieje) są wyrejestrowywane z domyślnego punktu subskrypcji.
- Jeśli zostanie podana opcja `DeregAll`, wszystkie subskrypcje (zgodne z właściwościami `< Topic>` i `< Filter >` , jeśli istnieją) zostaną wyrejestrowane ze wszystkich punktów subskrypcji.

Należy zauważyć, że nie można jawnie określić domyślnego punktu subskrypcji. Dlatego nie ma możliwości wyrejestrowania wszystkich subskrypcji tylko z tego punktu subskrypcji. Należy określić tematy.

SubIdentity (*MQPSC_SUBSCRIPTION_IDENTITY*)
Jest to łańcuch o zmiennej długości o maksymalnej długości 64 znaków. Jest on używany do reprezentowania aplikacji, która jest zainteresowana subskrypcją. Menedżer kolejek przechowuje zestaw tożsamości subskrybentów dla każdej subskrypcji. Każda subskrypcja może zezwalać na to, aby jej zestaw tożsamości przechowywał tylko jedną tożsamość lub nieograniczoną liczbę tożsamości.

Jeśli tożsamość `SubIdentity` znajduje się w zestawie tożsamości dla subskrypcji, jest ona usuwana z zestawu. Jeśli w wyniku tej operacji zestaw tożsamości stanie się pusty, subskrypcja zostanie usunięta z menedżera kolejek, chyba że jako wartość właściwości `RegOpt` określono wartość `LeaveOnly` . Jeśli zestaw tożsamości nadal zawiera inne tożsamości, subskrypcja nie zostanie usunięta z menedżera kolejek, a przepływ publikowania nie zostanie przerwany.

Jeśli zostanie podana opcja `SubIdentity` , ale tożsamość `SubIdentity` nie znajduje się w zestawie tożsamości dla subskrypcji, komenda **`Deregister Subscriber`** zakończy się niepowodzeniem z kodem powrotu *MQRCCF_SUB_IDENTITY_ERROR*.

Filtr (*MQPSC_FILTER*)
Wartością jest łańcuch określający filtr, który ma zostać wyrejestrowany. Musi on być dokładnie zgodny, włącznie z wielkością liter i wszystkimi obszarami, z filtrem subskrypcji, który został wcześniej zarejestrowany.

Tę właściwość można opcjonalnie powtórzyć, jeśli ma zostać wyrejestrowany więcej niż jeden filtr. Można ją pominąć, jeśli określono `< Topic>` lub jeśli w parametrze <RegOpt>określono wartość `DeregAll` .

Określone filtry mogą być podzbiorem tych zarejestrowanych, jeśli subskrybent chce zachować subskrypcje dla innych filtrów.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
Właściwość opcji rejestracji może przyjmować następujące wartości:

`DeregAll`
(*MQPSC_DEREGISTER_ALL*)

Wszystkie zgodne subskrypcje zarejestrowane dla tego subskrybenta mają zostać wyrejestrowane.

Jeśli zostanie podana opcja `DeregAll`:

- Można pominąć `< Topic>`, <SubPoint>i `< Filter >` .
- `< Topic>` i `< Filtr >` można powtórzyć, jeśli jest to wymagane.
- <SubPoint> nie może być powtarzany.

Jeśli **nie** zostanie podana opcja `DeregAll`:

- Należy podać `< Topic>` i w razie potrzeby można ją powtórzyć.
- Można pominąć <SubPoint> i `< filtr >` .
- <SubPoint> nie może być powtarzany.
- `< Filtr >` można powtórzyć, jeśli jest to wymagane.

Jeśli tematy i filtry są powtarzane, wszystkie subskrypcje zgodne ze wszystkimi kombinacjami tych dwóch elementów są usuwane. Na przykład komenda **`Deregister Subscriber`** , która określa trzy tematy i trzy filtry, podejmie próbę usunięcia dziewięciu subskrypcji.

`CorrelAs`
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Do identyfikacji subskrybenta używany jest identyfikator `CorrelId` w deskryptorze komunikatu (MQMD), który nie może mieć wartości zero. Musi być zgodna z identyfikatorem `CorrelId` użytym w oryginalnej subskrypcji.

`FullResp`
(*ODPOWIEDZI_MQPSC_FULL_RESPONSE*)

Jeśli określono opcję FullResp , wszystkie atrybuty subskrypcji są zwracane w komunikacie odpowiedzi, jeśli wykonanie komendy nie zakończy się niepowodzeniem.

Jeśli podano opcję FullResp , opcja DeregAll nie jest dozwolona w komendzie **Deregister Subscriber** . Nie jest również możliwe określenie wielu tematów. Wykonanie komendy kończy się niepowodzeniem z kodem powrotu *MQRCCF_REG_OPTIONS_ERROR*w obu przypadkach.

LeaveOnly
(*MQPSC_LEAVE_ONLY*)

W przypadku określenia tej opcji z tożsamością SubIdentity , która znajduje się w zestawie tożsamości dla subskrypcji, tożsamość SubIdentity jest usuwana z zestawu tożsamości dla subskrypcji. Subskrypcja nie jest usuwana z menedżera kolejek, nawet jeśli wynikowy zestaw tożsamości jest pusty. Jeśli wartość SubIdentity nie występuje w zestawie tożsamości, komenda kończy się niepowodzeniem z kodem powrotu *MQRCCF_SUB_IDENTITY_ERROR*.

Jeśli opcja LeaveOnly zostanie podana bez opcji SubIdentity, wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_REG_OPTIONS_ERROR*.

Jeśli nie zostanie podana opcja LeaveOnly ani opcja SubIdentity , subskrypcja zostanie usunięta niezależnie od zawartości zestawu tożsamości dla subskrypcji.

Brak
(MQPSC_NONE)

Wszystkie opcje przyjmują wartości domyślne. Ma to taki sam efekt, jak pominięcie właściwości opcji rejestracji. Jeśli w tym samym czasie zostaną podane inne opcje, opcja Brak zostanie zignorowana.

VariableUserIdentyfikator
(*ZMIENNA_MQPSC_ID_UŻYTKOWNIKA*)

Jeśli określono tożsamość subskrybenta (kolejka, menedżer kolejek i correlid) nie jest ograniczona do pojedynczego identyfikatora użytkownika. Różni się to od istniejącego zachowania menedżera kolejek, który wiąże identyfikator użytkownika oryginalnego komunikatu rejestracji z tożsamością subskrybenta, a następnie uniemożliwia używanie tej tożsamości przez innego użytkownika. Jeśli nowy subskrybent próbuje użyć tej samej tożsamości, zwracany jest kod powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Każdy użytkownik może modyfikować lub wyrejestrowywać subskrypcję, gdy ma odpowiednie uprawnienia, unikając sprawdzania, czy identyfikator użytkownika musi być zgodny z identyfikatorem oryginalnego subskrybenta.

Aby dodać tę opcję do istniejącej subskrypcji, komenda musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli subskrypcja, która ma zostać wyrejestrowana, ma ustawiony identyfikator VariableUserId , musi zostać ustawiona podczas wyrejestrowywania, aby wskazać, która subskrypcja jest wyrejestrowana. W przeciwnym razie do zidentyfikowania subskrypcji używany jest identyfikator użytkownika komendy **Deregister Subscriber** . Jest ona nadpisywana wraz z innymi identyfikatorami subskrybentów, jeśli podano nazwę subskrypcji.

Jeśli ta właściwość zostanie pominięta, domyślnie nie są ustawione żadne opcje rejestracji.

QMgrName (*MQPSC_Q_MGR_NAME*)
Wartością jest nazwa menedżera kolejek dla kolejki subskrybenta. Musi być ona zgodna z nazwą QMgrName używaną w oryginalnej subskrypcji.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa ReplyToQMgr w deskryptorze komunikatu (MQMD). Jeśli wynikowa nazwa jest pusta, wartością domyślną jest nazwa menedżera kolejek.

QName (*MQPSC_Q_NAME*)
Wartością jest nazwa kolejki subskrybenta. Musi być ona zgodna z nazwą QName używaną w oryginalnej subskrypcji.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa ReplyToQ w deskryptorze komunikatu (MQMD), która nie może być pusta.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Jeśli w komendzie **Deregister Subscriber** zostanie podana opcja SubName , wartość SubName ma pierwszeństwo przed wszystkimi innymi polami identyfikatora z wyjątkiem ID użytkownika, chyba że w samej subskrypcji ustawiono wartość VariableUserId . Jeśli parametr VariableUserId nie jest ustawiony, komenda **Deregister Subscriber** powiedzie się tylko wtedy, gdy ID użytkownika komunikatu komendy jest zgodny z ID użytkownika subskrypcji, jeśli komenda nie zakończy się niepowodzeniem z kodem powrotu *MQRCCF_DUPLICATE_IDENTITY*.

Jeśli istnieje subskrypcja, która jest zgodna z tradycyjną tożsamością tej komendy, ale nie ma SubName , komenda **Deregister Subscriber** kończy się niepowodzeniem z kodem powrotu *MQRCCF_SUB_NAME_ERROR*. Próba wyrejestrowania subskrypcji z nazwą SubName przy użyciu komunikatu komendy, który jest zgodny z tradycyjną tożsamością, ale bez nazwy SubName zakończy się powodzeniem.

SubUserDane (*MQPSC_SUBSCRIPTION_USER_DATA*)
Jest to łańcuch tekstowy o zmiennej długości. Wartość jest przechowywana przez menedżer kolejek z subskrypcją, ale nie ma wpływu na dostarczanie publikacji do subskrybenta. Wartość można zmienić, ponownie rejestrując się w tej samej subskrypcji z nową wartością. Ten atrybut jest przeznaczony dla aplikacji.

SubUserDane są zwracane w informacjach metatematu (MQCACF_REG_SUB_USER_DATA) dla subskrypcji, jeśli istnieją dane SubUser.

Przykład

Poniżej przedstawiono przykład danych NameValue dla komunikatu komendy **Deregister Subscriber** . W tym przykładzie przykładowa aplikacja wyrejestrowuje subskrypcję tematów, które zawierają najnowszy wynik dla wszystkich dopasowań. Tożsamość subskrybenta, w tym identyfikator CorrelId, jest pobierana z wartości domyślnych z deskryptora MQMD.

```
<psc>
  <Command>DeregSub</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Publikuj komunikat

Komunikat komendy **Publish** jest umieszczany w kolejce lub z menedżera kolejek do subskrybenta w celu opublikowania informacji o określonym temacie lub tematach.

Wymagane jest uprawnienie do umieszczania komunikatu w kolejce oraz uprawnienie do publikowania informacji dotyczących określonego tematu lub tematów.

Jeśli użytkownik ma uprawnienia do publikowania informacji o niektórych, ale nie wszystkich, tematach, do publikowania używane są tylko te tematy. Odpowiedź z ostrzeżeniem wskazuje, które tematy nie są używane do publikowania.

Jeśli subskrybent ma zgodne subskrypcje, menedżer kolejek przekazuje komunikat **Publish** do kolejek subskrybenta zdefiniowanych w odpowiednich komunikatach komendy **Register Subscriber** .

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD) wymaganych podczas wysyłania komunikatu komendy do menedżera kolejek i używanych, gdy menedżer kolejek przekazuje publikację do subskrybenta, zawiera sekcja Komunikat odpowiedzi menedżera kolejek .

Menedżer kolejek przekazuje komunikat **Publish** do innych menedżerów kolejek w sieci, które mają zgodne subskrypcje, chyba że jest to publikacja lokalna.

Dane publikacji, jeśli istnieją, są uwzględniane w treści komunikatu. Dane można opisać w folderze <mcd> w polu NameValueData nagłówka MQRFH2 .

Właściwości

Komenda (*MQPSC_COMMAND*)
Wartością jest `Publikuj` (*MQPSC_PUBLISH*).

Ta właściwość musi być określona.

Temat (*MQPSC_TOPIC*)
Wartość jest łańcuchem, który zawiera temat kategoryzujący tę publikację. Znaki wieloznaczne nie są dozwolone.

Należy dodać temat do listy nazw SYSTEM.QPUBSUB.QUEUE.NAMELIST, patrz sekcja Dodawanie strumienia , aby uzyskać instrukcje dotyczące wykonywania tego zadania.

Ta właściwość musi zostać określona i opcjonalnie może zostać powtórzona dla dowolnej liczby tematów.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Punkt subskrypcji, w którym publikacja jest publikowana.

W produkcie WebSphere Event Broker 6.0wartość właściwości <SubPoint> jest wartością atrybutu Punkt subskrypcji węzła publikowania, który obsługuje publikowanie.

W programie IBM WebSphere MQ 7.0.1wartość właściwości <SubPoint> musi być zgodna z nazwą punktu subskrypcji. Patrz sekcja Dodawanie punktu subskrypcji.

PubOpt (*MQPSC_PUBLIC OPTION*)
Właściwość opcji publikacji może przyjmować następujące wartości:

`RetainPub`
(*MQPSC_RETAIN_PUB*)

Menedżer kolejek zachowuje kopię publikacji. Jeśli ta opcja nie jest ustawiona, publikacja jest usuwana natychmiast po wysłaniu jej przez menedżer kolejek do wszystkich bieżących subskrybentów.

`IsRetainedPub`
(*MQPSC_IS_RETAINED_PUB*)

(Może być ustawiona tylko przez menedżera kolejek). Ta publikacja została zachowana przez menedżer kolejek. Menedżer kolejek ustawia tę opcję, aby powiadomić subskrybenta o tym, że ta publikacja została opublikowana wcześniej i zachowana, pod warunkiem, że subskrypcja została zarejestrowana z opcją `InformIfZachowana` . Jest ona ustawiana tylko w odpowiedzi na komunikat komendy `Zarejestruj subskrybenta` lub `Żądanie aktualizacji` . Zachowane publikacje, które są wysyłane bezpośrednio do subskrybentów, nie mają ustawionej tej opcji.

`Lokalna`
(*MQPSC_LOCAL*)

Ta opcja informuje menedżer kolejek, że ta publikacja nie może być wysyłana do innych menedżerów kolejek. Wszystkie subskrybenty zarejestrowane w tym menedżerze kolejek otrzymają tę publikację, jeśli mają zgodne subskrypcje.

`TylkoOtherSubs`
(*MQPSC_OTHER_SUBS_ONLY*)

Ta opcja umożliwia prostsze przetwarzanie aplikacji typu konferencyjnego, w których publikator jest również subskrybentem tego samego tematu. Informuje on menedżera kolejek, aby nie wysyłał publikacji do kolejki subskrybenta publikatora, nawet jeśli ma zgodną subskrypcję. Kolejka subskrybenta publikatora składa się z kolejki `QMgrName`, `QNamei` opcjonalnego identyfikatora `CorrelId`, zgodnie z opisem na poniższej liście.

`CorrelAs`
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Identyfikator `CorrelId` w strukturze MQMD (który nie może być zerowy) jest częścią kolejki subskrybenta publikatora w aplikacjach, w których publikator jest również subskrybentem.

Brak
(*MQPSC_NONE*)

Wszystkie opcje przyjmują wartości domyślne. Ma to taki sam efekt, jak pominięcie właściwości opcji publikacji. Jeśli w tym samym czasie zostaną podane inne opcje, opcja `Brak` zostanie zignorowana.

Można użyć więcej niż jednej opcji publikowania, wprowadzając dodatkowe elementy <PubOpt> .

Jeśli ta właściwość zostanie pominięta, domyślnie nie są ustawione żadne opcje publikacji.

PubTime (*MQPSC_PUBLISH_TIMESTAMP*)
Wartość jest opcjonalnym znacznikiem czasu publikacji ustawionym przez publikator. Ma długość 16 znaków i format:

```
YYYYMMDDHHMMSSTH
```

przy użyciu czasu uniwersalnego. Ta informacja nie jest sprawdzana przez menedżer kolejek przed wysłaniem do subskrybentów.

SeqNum (*MQPSC_SEQUENCE_NUMBER*)
Wartość jest opcjonalnym numerem kolejnym ustawionym przez publikator.

Wartość ta musi być zwiększana o 1 przy każdej publikacji. Nie jest to jednak sprawdzane przez menedżer kolejek, który jedynie przesyła te informacje do subskrybentów.

Jeśli publikacje dotyczące tego samego tematu są publikowane w różnych połączonych ze sobą menedżerach kolejek, to obowiązkiem publikatorów jest zapewnienie, aby numery kolejne, jeśli są używane, były znaczące.

QMgrName (*MQPSC_Q_MGR_NAME*)
Wartość jest łańcuchem zawierającym nazwę menedżera kolejek dla kolejki subskrybenta publikatora w aplikacjach, w których publikator jest również subskrybentem (patrz tylko sekcja `OtherSubs`).

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `ReplyToQMgr` w deskryptorze komunikatu (MQMD). Jeśli wynikowa nazwa jest pusta, wartością domyślną jest nazwa menedżera kolejek.

QName (*MQPSC_Q_NAME*)
Wartość jest łańcuchem zawierającym nazwę kolejki subskrybenta publikatora w aplikacjach, w których publikator jest również subskrybentem (patrz sekcja `TylkoOtherSubs`).

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `ReplyToQ` w deskryptorze komunikatu (MQMD), która nie może być pusta, jeśli ustawiona jest opcja `TylkoOtherSubs` .

Przykład

Poniżej znajdują się przykłady danych *NameValue* dla komunikatu komendy **`Publish`** .

Pierwszy przykład dotyczy publikacji wysyłanej przez symulator dopasowania w przykładowej aplikacji w celu wskazania, że zgodność została rozpoczęta.

```
<psc>
  <Command>Publish</Command>
  <Topic>Sport/Soccer/Event/MatchStarted</Topic>
</psc>
```

Drugi przykład dotyczy zachowanej publikacji. Publikowany jest najnowszy wynik w meczu między Team1 i Team2 .

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
```

```
   <Topic>Sport/Soccer/State/LatestScore/Team1 Team2</Topic>
 </psc>
```

Komunikat rejestrowania subskrybenta

Komunikat komendy **Register Subscriber** jest wysyłany do menedżera kolejek przez subskrybenta lub przez inną aplikację w imieniu subskrybenta, aby wskazać, że chce on subskrybować jeden lub więcej tematów w punkcie subskrypcji. Można również określić filtr treści komunikatu.

W wyrażeniach filtru publikowania/subskrypcji zagnieżdżanie nawiasów powoduje obniżenie wydajności w trybie wykładniczym. Unikaj zagnieżdżania nawiasów do głębokości większej niż około 6.

Komunikat jest wysyłany do systemu SYSTEM.BROKER.CONTROL.QUEUE, która jest kolejką sterującą menedżera kolejek. Oprócz uprawnień dostępu (ustawionych przez administratora systemu menedżera kolejek) do tematu lub tematów w subskrypcji wymagane jest uprawnienie do umieszczenia komunikatu w tej kolejce.

Jeśli użytkownik ma uprawnienia do niektórych, ale nie wszystkich, tematów, rejestrowane są tylko te, które mają uprawnienia. Ostrzeżenie wskazuje te, które nie są zarejestrowane.

Szczegółowe informacje na temat parametrów deskryptora komunikatu (MQMD) wymaganych podczas wysyłania komunikatu komendy do menedżera kolejek zawiera sekcja "Ustawienia MQMD w komunikatach komend dla menedżera kolejek" na stronie 927 .

Jeśli kolejka odpowiedzi jest tymczasową kolejką dynamiczną, po zamknięciu kolejki subskrypcja jest automatycznie wyrejestrowywana przez menedżer kolejek.

Właściwości

Komenda (*MQPSC_COMMAND*)
Wartością jest RegSub (*MQPSC_REGISTER_SUBSCRIBER*). Ta właściwość musi być określona.

Temat (*MQPSC_TOPIC*)
Temat, dla którego subskrybent chce otrzymywać publikacje. Znaki wieloznaczne mogą być określane jako część tematu.

Jeśli do sprawdzenia subskrypcji utworzonej w ten sposób używana jest komenda MQSC **display sub** , wartość znacznika < Topic> jest wyświetlana jako właściwość TOPICSTR subskrypcji.

Ta właściwość jest wymagana i może być opcjonalnie powtarzana dla dowolnej liczby tematów.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Wartością jest punkt subskrypcji, do którego jest przyłączona subskrypcja.

Jeśli ta właściwość zostanie pominięta, zostanie użyty domyślny punkt subskrypcji.

W programie WebSphere Event Broker 6.0wartość właściwości <SubPoint> musi być zgodna z wartością atrybutu Punkt subskrypcji węzłów publikowania, które zostały zasubskrybowane.

W programie IBM WebSphere MQ 7.0.1wartość właściwości <SubPoint> musi być zgodna z nazwą punktu subskrypcji. Patrz sekcja Dodawanie punktu subskrypcji.

Filtr (*MQPSC_FILTER*)
Wartość jest wyrażeniem SQL używanym jako filtr treści komunikatów publikowania. Jeśli publikacja w określonym temacie jest zgodna z filtrem, jest ona wysyłana do subskrybenta. Ta właściwość odpowiada łańcuchowi wyboru używanemu w wywołaniach MQSUB i MQOPEN. Więcej informacji na ten temat zawiera sekcja Wybieranie treści komunikatu .

Jeśli ta właściwość zostanie pominięta, filtrowanie treści nie będzie wykonywane.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
Ta właściwość opcji rejestracji może przyjmować następujące wartości:

AddName
(*MQPSC_ADD_NAME*)

Jeśli została określona dla istniejącej subskrypcji, która jest zgodna z tradycyjną tożsamością tej komendy rejestrowania subskrypcji, ale nie ma bieżącej wartości SubName , do subskrypcji jest dodawana nazwa SubName określona w tej komendzie.

Jeśli zostanie podany parametr AddName , pole SubName jest obowiązkowe. W przeciwnym razie zostanie zwrócony błąd MQRCCF_REG_OPTIONS_ERROR.

CorrelAs
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Identyfikator CorrelId w deskryptorze komunikatu (MQMD) jest używany podczas wysyłania zgodnych publikacji do kolejki subskrybenta. Wartość CorrelId nie może być równa zero.

FullResp
(*ODPOWIEDZI_MQPSC_FULL_RESPONSE*)

Jeśli podano wszystkie atrybuty subskrypcji, są one zwracane w komunikacie odpowiedzi, jeśli wykonanie komendy nie powiedzie się.

FullResp jest poprawny tylko wtedy, gdy komunikat komendy odwołuje się do pojedynczej subskrypcji. Dlatego w komendzie dozwolony jest tylko jeden temat. W przeciwnym razie wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_REG_OPTIONS_ERROR*.

InformIfRet
(*MQPSC_INFORM_IF_ZACHOWANY*)

Menedżer kolejek informuje subskrybenta o tym, czy publikacja jest zachowywana podczas wysyłania komunikatu publikowania w odpowiedzi na komunikat komendy **Register Subscriber** lub **Request Update** . W tym celu menedżer kolejek dołącza do komunikatu opcję publikacji IsRetainedPub .

JoinExcl
(*MQPSC_JOIN_EXCLUSIVE*)

Ta opcja wskazuje, że podana tożsamość SubIdentity powinna zostać dodana jako wyłączny element zestawu tożsamości dla subskrypcji i że do zestawu nie można dodać żadnych innych tożsamości.

Jeśli tożsamość już dołączyła do 'shared' i jest jedyną pozycją w zestawie, zestaw jest zmieniany na blokadę na wyłączność posiadaną przez tę tożsamość. W przeciwnym razie, jeśli subskrypcja ma obecnie inne tożsamości w zestawie tożsamości (z dostępem współużytkowanym), wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_SUBSCRIPTION_IN_USE*.

JoinShared
(*MQPSC_JOIN_SHARED*)

Ta opcja wskazuje, że podana tożsamość SubIdentity powinna zostać dodana do zestawu tożsamości dla subskrypcji.

Jeśli subskrypcja jest obecnie zablokowana na wyłączność (przy użyciu opcji JoinExcl), komenda kończy się niepowodzeniem z kodem powrotu *MQRCCF_SUBSCRIPTION_LOCKED*, chyba że tożsamość, która ma zablokowaną subskrypcję, jest taka sama jak tożsamość w tym komunikacie komendy. W takim przypadku blokada jest automatycznie modyfikowana na blokadę ze współużytkowaniem.

Lokalna
(*MQPSC_LOCAL*)

Subskrypcja jest lokalna i nie jest dystrybuowana do innych menedżerów kolejek w sieci. Publikacje wykonane w innych menedżerach kolejek nie są dostarczane do tego subskrybenta, chyba że ma on również odpowiednią subskrypcję globalną.

TylkoNewPubs
(*MQPSC_NEW_PUBS_ONLY*)

Zachowane publikacje istniejące w momencie rejestrowania subskrypcji nie są wysyłane do subskrybenta; wysyłane są tylko nowe publikacje.

Jeśli subskrybent zarejestruje się ponownie i zmieni tę opcję w taki sposób, że nie jest już ustawiona, może zostać ponownie wysłana publikacja, która została już do niego wysłana.

NoAlter
(*MQPSC_NO_ALTER*)

Atrybuty istniejącej zgodnej subskrypcji nie są zmieniane.

Podczas tworzenia subskrypcji ta opcja jest ignorowana. Wszystkie inne określone opcje mają zastosowanie do nowej subskrypcji.

Jeśli tożsamość `SubIdentity` ma również jedną z opcji łączenia (`JoinExcl` lub `JoinShared`) Tożsamość jest dodawana do zestawu tożsamości niezależnie od tego, czy określono opcję `NoAlter` .

Brak
(*MQPSC_NONE*)

Wszystkie opcje rejestracji przyjmują wartości domyślne.

Jeśli subskrybent jest już zarejestrowany, jego opcje są resetowane do wartości domyślnych (należy zauważyć, że nie ma to takiego samego wpływu, jak pominięcie właściwości opcji rejestracji), a utrata ważności subskrypcji jest aktualizowana na podstawie deskryptora MQMD komunikatu **`Register Subscriber`** .

Jeśli w tym samym czasie zostaną podane inne opcje rejestracji, opcja `Brak` zostanie zignorowana.

NonPers
(*MQPSC_NON_PERSISTENT*)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta jako komunikaty nietrwałe.

Pers
(*MQPSC_PERSISTENT*)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta jako komunikaty trwałe.

PersAsPub
(*MQPSC_PERSISTENT_AS_PUBLISH*)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta z trwałością określoną przez publikator. To jest zachowanie domyślne.

KolejkaPersAs
(*MQPSC_PERSISTENT_AS_Q*)

Publikacje zgodne z tą subskrypcją są dostarczane do subskrybenta z trwałością określoną w kolejce subskrybenta.

PubOnReqOnly
(*MQPSC_PUB_ON_REQUEST_ONLY*)

Menedżer kolejek nie wysyła publikacji do subskrybenta, z wyjątkiem odpowiedzi na komunikat komendy **`Request Update`** .

VariableUserIdentyfikator
(*ZMIENNA_MQPSC_ID_UŻYTKOWNIKA*)

Jeśli określono tożsamość subskrybenta (kolejka, menedżer kolejek i correlid) nie jest ograniczona do pojedynczego identyfikatora użytkownika. Różni się to od istniejącego zachowania menedżera kolejek, który wiąże identyfikator użytkownika oryginalnego komunikatu rejestracji z tożsamością subskrybenta, a następnie uniemożliwia używanie tej tożsamości przez innego użytkownika. Jeśli nowy subskrybent próbuje użyć tej samej tożsamości, zwracana jest wartość *MQRCCF_DUPLICATE_SUBSCRIPTION* .

Umożliwia to każdemu użytkownikowi modyfikowanie lub wyrejestrowywanie subskrypcji, jeśli użytkownik ma odpowiednie uprawnienia. Dlatego nie ma potrzeby sprawdzania, czy identyfikator użytkownika jest zgodny z identyfikatorem pierwotnego subskrybenta.

Aby dodać tę opcję do istniejącej subskrypcji, komenda musi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli subskrypcja komendy **Request Update** ma ustawiony identyfikator VariableUserId , musi być on ustawiony w czasie aktualizacji żądania, aby wskazać, do której subskrypcji się odwołuje. W przeciwnym razie do zidentyfikowania subskrypcji używany jest identyfikator użytkownika komendy **Request Update** . Jest ona nadpisywana wraz z innymi identyfikatorami subskrybentów, jeśli podano nazwę subskrypcji.

Jeśli komunikat komendy **Register Subscriber** bez tego zestawu opcji odwołuje się do istniejącej subskrypcji, która ma ten zestaw opcji, opcja zostanie usunięta z tej subskrypcji, a identyfikator użytkownika subskrypcji zostanie naprawiony. Jeśli istnieje już subskrybent o takiej samej tożsamości (kolejka, menedżer kolejek i identyfikator korelacji), ale z innym ID użytkownika, który jest z nim powiązany, wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_DUPLICATE_IDENTITY* , ponieważ z tożsamością subskrybenta może być powiązany tylko jeden ID użytkownika.

Jeśli właściwość opcji rejestracji zostanie pominięta, a subskrybent jest już zarejestrowany, jego opcje rejestracji nie zostaną zmienione, a utrata ważności subskrypcji zostanie zaktualizowana na podstawie deskryptora MQMD komunikatu **Register Subscriber** .

Jeśli subskrybent nie jest jeszcze zarejestrowany, tworzona jest nowa subskrypcja, w której wszystkie opcje rejestracji przyjmują wartości domyślne.

Wartości domyślne to PersAsPub i nie są ustawione żadne inne opcje.

QMgrName (*MQPSC_Q_MGR_NAME*)
Wartością jest nazwa menedżera kolejek dla kolejki subskrybenta, do której menedżer kolejek wysyła zgodne publikacje.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa ReplyToQMgr w deskryptorze komunikatu (MQMD). Jeśli nazwa wynikowa jest pusta, domyślnie przyjmowana jest nazwa menedżera kolejek QMgrName.

QName (*MQPSC_Q_NAME*)
Wartością jest nazwa kolejki subskrybenta, do której menedżer kolejek wysyła zgodne publikacje.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa ReplyToQ w deskryptorze komunikatu (MQMD), która w tym przypadku nie może być pusta.

Jeśli kolejka jest tymczasową kolejką dynamiczną, nietrwałe dostarczanie publikacji (NonPers) musi być określona we właściwości <RegOpt> .

Jeśli kolejka jest tymczasową kolejką dynamiczną, po zamknięciu kolejki subskrypcja jest automatycznie wyrejestrowywana przez menedżer kolejek.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Jest to nazwa nadana konkretnej subskrypcji. Aby odwołać się do subskrypcji, można jej użyć zamiast menedżera kolejek, kolejki i opcjonalnego identyfikatora correlId .

Jeśli istnieje już subskrypcja z tą **SubName** , wszystkie inne atrybuty subskrypcji (Topic, QMgrName, QName, CorrelId, UserId, RegOpts, UserSubData i Expiry) zostaną przesłonięte atrybutami, które zostały przekazane w nowym komunikacie komendy Zarejestruj subskrybenta . Jeśli jednak opcja **SubName** zostanie użyta bez określonego pola QName i w nagłówku MQMD zostanie podana wartość Q ReplyTo, kolejka subskrybenta zostanie zmieniona na wartość Q ReplyTo.

Jeśli subskrypcja, która jest zgodna z tradycyjną tożsamością tej komendy, już istnieje, ale nie ma **SubName** , wykonanie komendy rejestracji nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION*, chyba że zostanie podana opcja **AddName** .

Próba zmodyfikowania istniejącej nazwanej subskrypcji przy użyciu innej komendy Zarejestruj subskrybenta , która określa tę samą **SubName** , a wartości Topic, QMgrName, QName i CorrelId

w nowej komendzie są zgodne z inną istniejącą subskrypcją, ze zdefiniowaną opcją SubName lub bez niej, zakończy się niepowodzeniem z kodem powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION*. Zapobiega to występowaniu dwóch nazw subskrypcji odwołujących się do tej samej subskrypcji.

SubIdentity (*MQPSC_SUBSCRIPTION_IDENTITY*)
Ten łańcuch jest używany do reprezentowania aplikacji zainteresowanej subskrypcją. Jest to łańcuch znaków o zmiennej długości, o maksymalnej długości 64 znaków i jest opcjonalny. Menedżer kolejek przechowuje zestaw tożsamości subskrybentów dla każdej subskrypcji. Każda subskrypcja może zezwalać, aby jej zestaw tożsamości zawierał tylko jedną tożsamość lub nieograniczoną liczbę tożsamości (patrz opcje **JoinShared** i **JoinExcl**).

Komenda subskrypcji, która określa opcję **JoinShared** lub **JoinExcl** , dodaje tożsamość **SubIdentity** do zestawu tożsamości subskrypcji, jeśli jeszcze jej nie ma i jeśli istniejący zestaw tożsamości zezwala na takie działanie. Oznacza to, że żaden inny subskrybent nie dołączył na wyłączność lub zestaw tożsamości jest pusty.

Każda zmiana atrybutów subskrypcji w wyniku wykonania komendy `Zarejestruj subskrybenta` , w której określono tożsamość **SubIdentity** , powiedzie się tylko wtedy, gdy będzie ona jedynym elementem zestawu tożsamości dla tej subskrypcji. W przeciwnym razie wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_SUBSCRIPTION_IN_USE*. Zapobiega to zmianie atrybutów subskrypcji bez uwzględniania innych zainteresowanych subskrybentów.

Jeśli zostanie podany łańcuch znaków dłuższy niż 64 znaki, wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_SUB_IDENTITY_ERROR*.

SubUserDane (*MQPSC_SUBSCRIPTION_USER_DATA*)
Jest to łańcuch tekstowy o zmiennej długości. Wartość jest przechowywana przez menedżer kolejek z subskrypcją, ale nie ma wpływu na dostarczanie publikacji do subskrybenta. Wartość można zmienić, ponownie rejestrując się w tej samej subskrypcji z nową wartością. Ten atrybut jest przeznaczony do użycia przez aplikację.

Dane **SubUser** są zwracane w informacjach metatematu (*MQCACF_REG_SUB_USER_DATA*) dla subskrypcji, jeśli istnieją.

Jeśli zostanie podana więcej niż jedna wartość opcji rejestracji `NonPers, PersAsPub, PersAsQueue, and Pers`, zostanie użyta tylko ostatnia z nich. Nie można łączyć tych opcji w ramach pojedynczej subskrypcji.

Przykład

Poniżej przedstawiono przykład danych `NameValue` dla komunikatu komendy **Register Subscriber** . W przykładowej aplikacji usługa wyników używa tego komunikatu do zarejestrowania subskrypcji tematów zawierających najnowsze wyniki we wszystkich dopasowaniach z ustawioną opcją "Trwałe jako publikowanie". Tożsamość subskrybenta, w tym identyfikator `CorrelId`, jest pobierana z wartości domyślnych z deskryptora MQMD.

```
<psc>
  <Command>RegSub</Command>
  <RegOpt>PersAsPub</RegOpt>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Komunikat żądania aktualizacji

Komunikat komendy **Request Update** jest wysyłany z subskrybenta do menedżera kolejek w celu zażądania bieżących zachowanych publikacji dla określonego tematu i punktu subskrypcji, które są zgodne z podanym (opcjonalnym) filtrem.

Ten komunikat jest wysyłany do systemu *SYSTEM.BROKER.CONTROL.QUEUE*-kolejka sterująca menedżera kolejek. Oprócz uprawnień dostępu do tematu w żądaniu aktualizacji wymagane jest uprawnienie do

umieszczenia komunikatu w tej kolejce. Uprawnienia te są ustawiane przez administratora systemu menedżera kolejek.

Ta komenda jest zwykle używana, jeśli subskrybent określił opcję PubOnReqOnly podczas rejestrowania. Jeśli menedżer kolejek ma jakiekolwiek zgodne zachowane publikacje, są one wysyłane do subskrybenta. Jeśli menedżer kolejek nie ma zgodnych zachowanych publikacji, żądanie kończy się niepowodzeniem z kodem powrotu *MQRCCF_NO_RETAINED_MSG*. Requester musi mieć wcześniej zarejestrowaną subskrypcję z tymi samymi wartościami Topic, SubPointi Filter.

Właściwości

Komenda (*MQPSC_COMMAND*)
Wartością jest ReqUpdate (*MQPSC_REQUEST_UPDATE*). Ta właściwość musi być określona.

Temat (*MQPSC_TOPIC*)
Wartością jest temat, którego żąda subskrybent. Znaki wieloznaczne są dozwolone.

Ta właściwość musi zostać określona, ale w tym komunikacie dozwolone jest tylko jedno wystąpienie.

SubPoint (*MQPSC_SUBSCRIPTION_POINT*)
Wartością jest punkt subskrypcji, do którego jest przyłączona subskrypcja.

Jeśli ta właściwość zostanie pominięta, zostanie użyty domyślny punkt subskrypcji.

Filtr (*MQPSC_FILTER*)
Wartością jest wyrażenie ESQL używane jako filtr treści komunikatów publikacji. Jeśli publikacja w określonym temacie jest zgodna z filtrem, jest ona wysyłana do subskrybenta.

Właściwość < Filtr > powinna mieć taką samą wartość, jak określona w oryginalnej subskrypcji, dla której jest teraz żądana aktualizacja.

Jeśli ta właściwość zostanie pominięta, filtrowanie treści nie będzie wykonywane.

RegOpt (*MQPSC_REGISTRATION_OPTION*)
Właściwość opcji rejestracji może przyjmować następującą wartość:

CorrelAs
(*MQPSC_CORREL_ID_AS_IDENTITY*)

Parametr CorrelId w deskryptorze komunikatu (MQMD), który nie może mieć wartości zero, jest używany podczas wysyłania zgodnych publikacji do kolejki subskrybenta.

Brak
(*MQPSC_NONE*)

Wszystkie opcje przyjmują wartości domyślne. Efekt jest taki sam, jak pominięcie właściwości <RegOpt> . Jeśli w tym samym czasie zostaną podane inne opcje, opcja Brak zostanie zignorowana.

VariableUserIdentyfikator
(*ZMIENNA_MQPSC_ID_UŻYTKOWNIKA*)

Jeśli określono tożsamość subskrybenta (kolejka, menedżer kolejek i correlid) nie jest ograniczona do pojedynczego identyfikatora użytkownika. Różni się to od istniejącego zachowania menedżera kolejek, który wiąże identyfikator użytkownika oryginalnego komunikatu rejestracji z tożsamością subskrybenta, a następnie uniemożliwia używanie tej tożsamości przez innego użytkownika. Jeśli nowy subskrybent próbuje użyć tej samej tożsamości, komenda kończy się niepowodzeniem z kodem powrotu *MQRCCF_DUPLICATE_SUBSCRIPTION*.

Dzięki temu każdy użytkownik może modyfikować lub wyrejestrowywać subskrypcję, gdy ma odpowiednie uprawnienia. Dlatego nie ma potrzeby sprawdzania, czy identyfikator użytkownika jest zgodny z identyfikatorem oryginalnego subskrybenta.

Aby dodać tę opcję do istniejącej subskrypcji, komenda musi pochodzić z tego samego identyfikatora użytkownika, co pierwotna subskrypcja.

Jeśli subskrypcja komendy **Request Update** ma ustawiony identyfikator `VariableUserId` , musi być on ustawiony w czasie aktualizacji żądania, aby wskazać, do której subskrypcji się odwołuje. W przeciwnym razie do zidentyfikowania subskrypcji używany jest identyfikator użytkownika komendy **Request Update** . Jest ona nadpisywana wraz z innymi identyfikatorami subskrybentów, jeśli podano nazwę subskrypcji.

Jeśli ta właściwość zostanie pominięta, domyślnie nie są ustawione żadne opcje rejestracji.

QMgrName (*MQPSC_Q_MGR_NAME*)
Wartość jest nazwą menedżera kolejek dla kolejki subskrybenta, do której menedżer kolejek wysyła zgodną zachowaną publikację.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `ReplyToQMgr` w deskryptorze komunikatu (MQMD). Jeśli nazwa wynikowa jest pusta, domyślnie przyjmowana jest nazwa menedżera kolejek `QMgrName`.

QName (*MQPSC_Q_NAME*)
Wartością jest nazwa kolejki subskrybenta, do której menedżer kolejek wysyła zgodną zachowaną publikację.

Jeśli ta właściwość zostanie pominięta, wartością domyślną jest nazwa `ReplyToQ` w deskryptorze komunikatu (MQMD), która w tym przypadku nie może być pusta.

SubName (*MQPSC_SUBSCRIPTION_NAME*)
Jest to nazwa nadana konkretnej subskrypcji. Wartość SubName podana w komendzie **Request Update** ma pierwszeństwo przed wszystkimi innymi polami identyfikatora z wyjątkiem ID użytkownika, chyba że w samej subskrypcji ustawiono wartość `VariableUserId` . Jeśli identyfikator `VariableUserId` nie jest ustawiony, komenda *Request Update* powiedzie się tylko wtedy, gdy identyfikator użytkownika komunikatu komendy jest zgodny z identyfikatorem użytkownika subskrypcji. Jeśli identyfikator użytkownika komunikatu komendy nie jest zgodny z identyfikatorem użytkownika subskrypcji, komenda nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_DUPLICATE_IDENTITY*.

Jeśli identyfikator `VariableUserId` jest ustawiony, a identyfikator użytkownika różni się od identyfikatora użytkownika subskrypcji, komenda powiedzie się, jeśli identyfikator użytkownika nowego komunikatu komendy ma uprawnienie do przeglądania kolejki strumienia i umieszczania w kolejce subskrybenta subskrypcji. W przeciwnym razie wykonanie komendy nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_NOT_AUTHORIZED*.

Jeśli istnieje subskrypcja, która jest zgodna z tradycyjną tożsamością tej komendy, ale nie ma SubName, wykonanie komendy **Request Update** nie powiedzie się i zostanie zwrócony kod powrotu *MQRCCF_SUB_NAME_ERROR*.

Jeśli zostanie podjęta próba zażądania aktualizacji dla subskrypcji, która ma SubName , przy użyciu komunikatu komendy zgodnego z tradycyjną tożsamością, ale bez określonej SubName , komenda powiedzie się.

Przykład

Poniżej przedstawiono przykład danych `NameValue` dla komunikatu komendy **Request Update** . W przykładowej aplikacji usługa wyników używa tego komunikatu do zażądania zachowanych publikacji zawierających najnowsze wyniki dla wszystkich zespołów. Tożsamość subskrybenta, w tym identyfikator `CorrelId`, jest pobierana z wartości domyślnych z deskryptora MQMD.

```
<psc>
  <Command>ReqUpdate</Command>
  <RegOpt>CorrelAsId</RegOpt>
  <Topic>Sport/Soccer/State/LatestScore/#</Topic>
</psc>
```

Komunikat odpowiedzi menedżera kolejek

Komunikat **Queue Manager Response** jest wysyłany z menedżera kolejek do kolejki ReplyToQ publikatora lub subskrybenta w celu wskazania powodzenia lub niepowodzenia komunikatu komendy odebranego przez menedżer kolejek, jeśli deskryptor komunikatu komendy określa, że odpowiedź jest wymagana.

Komunikat odpowiedzi jest zawarty w polu NameValueData nagłówka MQRFH2 w folderze <pscr> .

W przypadku ostrzeżenia lub błędu komunikat odpowiedzi zawiera folder <psc> z komunikatu komendy oraz folder <pscr> . Dane komunikatu (jeśli istnieją) nie są zawarte w komunikacie odpowiedzi menedżera kolejek. W przypadku błędu żaden komunikat, który spowodował błąd, nie został przetworzony. W przypadku ostrzeżenia część komunikatu mogła zostać przetworzona pomyślnie.

Jeśli wystąpi błąd podczas wysyłania odpowiedzi:

- W przypadku komunikatów publikowania menedżer kolejek próbuje wysłać odpowiedź do kolejki niedostarczonych komunikatów IBM MQ , jeśli operacja MQPUT nie powiedzie się. Umożliwia to wysyłanie publikacji do subskrybentów nawet wtedy, gdy odpowiedzi nie można wysłać z powrotem do publikatora.
- W przypadku innych komunikatów lub jeśli nie można wysłać odpowiedzi publikowania do kolejki niedostarczonych komunikatów, rejestrowany jest błąd, a komunikat komendy jest zwykle wycofywany. To, czy tak się stanie, zależy od sposobu skonfigurowania węzła MQInput.

Właściwości

Zakończenie (*MQPSCR_COMPLETION*)
Kod zakończenia, który może przyjmować jedną z trzech wartości:

OK
Komenda została zakończona pomyślnie

ostrzeżenie
Komenda została zakończona, ale z ostrzeżeniem

Błąd
Błąd komendy

Odpowiedź (*MQPSCR_RESPONSE*)
Odpowiedź na komunikat komendy, jeśli komenda wygenerowała kod zakończenia ostrzeżenie lub błąd. Zawiera właściwość < Reason> i może zawierać inne właściwości wskazujące przyczynę ostrzeżenia lub błędu.

W przypadku jednego lub większej liczby błędów istnieje tylko jeden folder odpowiedzi, który wskazuje przyczynę tylko pierwszego błędu. W przypadku jednego lub większej liczby ostrzeżeń dla każdego ostrzeżenia istnieje folder odpowiedzi.

Przyczyna (*MQPSCR_REASON*)
Kod przyczyny kwalifikujący kod zakończenia, jeśli kod zakończenia jest ostrzeżeniem lub błędem. Jest on ustawiony na jeden z kodów błędów wymienionych w poniższym przykładzie. Właściwość < Reason> znajduje się w folderze < Response> . Po kodzie przyczyny może następować dowolna poprawna właściwość z folderu <psc> (na przykład nazwa tematu) wskazująca przyczynę błędu lub ostrzeżenia. Jeśli pojawi się kod przyczyny? ???, Sprawdź poprawność danych, na przykład dopasowując nawiasy kątowe (< >).

Przykłady

Poniżej przedstawiono przykłady danych NameValue w komunikacie **Queue Manager Response** . Pomyślna odpowiedź może być następująca:

```
<pscr>
  <Completion>ok</Completion>
</pscr>
```

Poniżej przedstawiono przykład odpowiedzi dotyczącej niepowodzenia. Błąd jest błędem filtru. Pierwszy łańcuch NameValueData zawiera odpowiedź, drugi zawiera oryginalną komendę.

```
<pscr>
  <Completion>error</Completion>
  <Response>
    <Reason>3150</Reason>
  </Reponse>
</pscr>

<psc>
 ...
 command message (to which
 the queue manager is responding)
 ...
</psc>
```

Poniżej przedstawiono przykład ostrzeżenia (z powodu nieautoryzowanych tematów). Pierwszy łańcuch NameValueData zawiera odpowiedź, drugi łańcuch NameValueData zawiera oryginalną komendę.

```
<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>
```

Kody przyczyny publikowania/subskrypcji

Te kody przyczyny mogą zostać zwrócone w polu Przyczyna folderu odpowiedzi publikowania/subskrypcji <pscr> . Wymienione są również stałe, które mogą być używane do reprezentowania tych kodów w językach programowania C lub C + +.

Stałe MQRC_ wymagają pliku nagłówkowego produktu IBM MQ cmqc.h . Stałe MQRCCF_ wymagają pliku nagłówkowego IBM MQ cmqcfc.h (oprócz *MQRCCF_FILTER_ERROR* i *MQRCCF_WRONG_USER*, które wymagają pliku nagłówkowego cmqpsc.h).

Kod przyczyny i tekst	Objaśnienie	Wystawiony przez
2336 BŁĄD MQRC_RFH_COMMAND_ERROR	Poprawne wartości pola < Command> folderu <psc> to: RegSub, DeregSub, Publish, DeletePubi ReqUpdate. Każda inna wartość powoduje wygenerowanie tego kodu błędu.	Dowolna komenda
2337 BŁĄD ANALIZOWANIA MQRC_RFH_	Foldery <psc> i <mcd> mają zestaw poprawnych parametrów, które można w nich określić. Sprawdź opisy tych folderów i upewnij się, że nie podano niepoprawnych parametrów.	Dowolna komenda

Kod przyczyny i tekst	Objaśnienie	Wystawiony przez
2338 MQRC_RFH_DUPLICATE_PARM	Niektóre parametry (na przykład Topic) w folderze <psc> mogą być powtarzane, ale inne (na przykład Command) nie mogą być powtarzane. Sprawdź, czy nie został zduplikowany parametr niepowtarzalny.	Dowolna komenda
2339 BRAK MQRC_RFH_PARM_MISSING	Niektóre parametry w folderach <psc> lub <mcd> są opcjonalne i można je pominąć; niektóre z nich są obowiązkowe i nie można ich pominąć. Sprawdź, czy w folderach <psc> i <mcd> zostały uwzględnione wszystkie obowiązkowe parametry.	Dowolna komenda
2551 MQRC_SELECTION_NOT_AVAILABLE	Brak dostępnego dostawcy rozszerzonego wyboru komunikatów w celu określenia, którzy subskrybenci z określonym filtrem powinni otrzymać publikację.	Publikowanie, rejestrowanie subskrybenta i żądanie aktualizacji
	Brak dostępnego dostawcy rozszerzonego wyboru komunikatów do obsługi filtru określonego subskrybenta.	Zarejestruj subskrybent i zażądaj aktualizacji
2554 BŁĄD MQRC_CONTENT_ERROR	Dostawca rozszerzonego wyboru komunikatów wykrył błąd w bieżącej lub zachowanej publikacji.	Publikowanie i żądanie aktualizacji
3008 MQRCCF_COMMAND_FAILED	Wystąpił błąd wewnętrzny, który uniemożliwił poprawne wykonanie komendy. Ten błąd może wystąpić, jeśli komenda zostanie ponownie uruchomiona. Dziennik zdarzeń systemowych dla menedżera kolejek zawiera informacje, które powinny być używane podczas zgłaszania problemu do IBM.	Dowolna komenda
3072 BŁĄD MQRCCF_TOPIC_ERROR	Co najmniej jedna z wartości podanych dla parametru Topic jest niepoprawna. Sprawdź, czy wartości tematu są zgodne z określonymi ograniczeniami.	Dowolna komenda
3073 MQRCCF_NOT_REGISTERED (niezarejestrowane)	Kombinacja opcji SubPoint, Topic i Filter określonych w komendzie DeregSub lub ReqUpdate nie była kombinacją poprzednio zarejestrowanej lub, w przypadku komendy DeregSub , jeśli podano opcję DeregAll , jedna z właściwości SubPoint, Topic lub Filter nie została użyta do wyrejestrowania żadnej subskrypcji.	Komendy wyrejestrowywania subskrybenta i żądania aktualizacji

Kod przyczyny i tekst	Objaśnienie	Wystawiony przez
3074 BŁĄD MQRCCF_Q_MGR_NAME_ERROR	Określony menedżer kolejek jest niepoprawny lub nie jest dostępny albo nie istnieje.	Komendy wyrejestrowywania subskrybenta, publikowania, rejestrowania subskrybenta i żądania aktualizacji
3076 BŁĄD MQRCCF_Q_NAME_ERROR	Podana nazwa kolejki jest niepoprawna lub kolejka nie istnieje w określonym menedżerze kolejek.	Komendy wyrejestrowywania subskrybenta, publikowania, rejestrowania subskrybenta i żądania aktualizacji
3077 MQRCCF_NO_RETAINED_MSG	Brak zachowanych komunikatów dla określonego tematu. Może to być błąd lub nie, w zależności od projektu aplikacji.	Komenda żądania aktualizacji
3079 MQRCCF_INCORRECT_Q	Komendy RegSub, DeregSubi ReqUpdate są zawsze wysyłane do systemu SYSTEM.BROKER.CONTROL.QUEUE menedżera kolejek, dla którego są przeznaczone. Komendy publikowania i usuwania publikacji są wysyłane do kolejki wejściowej dla konkretnego przepływu komunikatów publikowania/subskrypcji, dla którego są przeznaczone. Jest to określane podczas projektowania przepływu komunikatów. Ten kod błędu jest zwracany, jeśli komenda zostanie wysłana do niewłaściwej kolejki.	Dowolna komenda
3080 BŁĄD MQRCCF_CORREL_ID_ERROR	Jako jeden z parametrów RegOpt podano identyfikator CorrelAs. Jednak pole CorrelId deskryptora MQMD nie zawiera poprawnego identyfikatora korelacji (tzn. ma wartość MQCI_NONE).	Komendy wyrejestrowywania subskrybenta i rejestrowania subskrybenta
3081 MQRCCF_NOT_AUTHORIZED (nieautoryzowany)	Brak autoryzacji do wykonania żądanej akcji. Ustawienia autoryzacji dla menedżera kolejek są obsługiwane przez administratora systemu za pomocą edytora hierarchii tematów.	Komendy publikowania i rejestrowania subskrybenta
3083 BŁĄD MQRCCF_REG_OPTIONS_ERROR	W folderze <psc> podano nierozpoznany parametr RegOpt , który zawiera komendę RegSub lub DeregSub .	Komendy wyrejestrowywania subskrybenta i rejestrowania subskrybenta

Kod przyczyny i tekst	Objaśnienie	Wystawiony przez
3084 MQRCCF_PUB_OPTIONS_ERROR	W folderze <psc> zawierającym komendę publikowania podano nierozpoznany parametr PubOpt .	Komenda publikowania
3087 BŁĄD MQRCCF_DEL_OPTIONS_ERROR	Podano nierozpoznany parametr DelOpt w folderze <psc> , który zawiera komendę DeletePub .	Komenda usuwania publikacji
3150 MQRCCF_FILTER_BŁĄD	Wartość określona dla parametru Filtr jest niepoprawna. Sprawdź sekcję opisującą poprawną składnię wyrażeń filtru i upewnij się, że wyrażenie jest zgodne.	Komendy wyrejestrowywania subskrybenta, rejestrowania subskrybenta i żądania aktualizacji
3151 MQRCCF_WRONG_USER	Istnieje już subskrypcja zgodna z określoną, ale została zarejestrowana przez innego użytkownika. Subskrypcja może zostać zmieniona lub wyrejestrowana tylko przez użytkownika, który ją pierwotnie zarejestrował.	Komendy wyrejestrowywania subskrybenta, rejestrowania subskrybenta i żądania aktualizacji
3152 MQRCCF_DUPLICATE_SUBSCRIPTION	Istnieje już zgodna subskrypcja o innej nazwie.	
3153 BŁĄD MQRCCF_SUB_NAME_ERROR	Format nazwy subskrypcji jest niepoprawny lub istnieje już zgodna subskrypcja bez nazwy subskrypcji.	
3154 BŁĄD MQRCCF_SUB_IDENTITY_ERROR	Wystąpił błąd parametru tożsamości subskrypcji. Podana wartość przekracza maksymalną dozwoloną długość lub tożsamość subskrypcji nie jest obecnie elementem zestawu tożsamości subskrypcji i nie określono opcji rejestracji łączenia.	
3155 MQRCCF_SUBSCRIPTION_IN_USE	Podjęto próbę zmodyfikowania lub wyrejestrowania subskrypcji przez element zestawu tożsamości, który nie był jedynym elementem tego zestawu.	
3156 MQRCCF_SUBSCRIPTION_LOCKED (zablokowana subskrypcja MQ)	Subskrypcja jest obecnie na wyłączność zablokowana przez inną tożsamość.	
3157 MQRCCF_ALREADY_JOINED	Podano opcję rejestracji łączenia, ale tożsamość subskrybenta była już elementem zestawu tożsamości subskrypcji.	

Ustawienia MQMD w komunikatach komend dla menedżera kolejek

Aplikacje, które wysyłają komunikaty komend do menedżera kolejek, używają następujących ustawień pól w deskryptorze komunikatu (MQMD). W tym miejscu nie są wyświetlane pola, które pozostają wartością domyślną lub mogą być ustawione na dowolną poprawną wartość w zwykły sposób.

Raport
Patrz MsgType i CorrelId.

MsgType
Dla pola MsgType należy ustawić wartość *MQMT_REQUEST* lub *MQMT_DATAGRAM*. Jeśli wartość MsgType nie jest ustawiona na jedną z tych wartości, zwracana jest wartość *MQRC_MSG_TYPE_ERROR* .

Dla komunikatu komendy MsgType należy ustawić wartość *MQMT_REQUEST* , jeśli odpowiedź jest zawsze wymagana. Flagi MQRO_PAN i MQRO_NAN w polu Raport nie są w tym przypadku istotne.

Jeśli wartość MsgType jest ustawiona na *MQMT_DATAGRAM*, odpowiedzi zależą od ustawienia flag MQRO_PAN i MQRO_NAN w polu Raport :

- Tylko MQRO_PAN oznacza, że menedżer kolejek wysyła odpowiedź tylko wtedy, gdy wykonanie komendy powiedzie się.
- Sam MQRO_NAN oznacza, że menedżer kolejek wysyła odpowiedź tylko wtedy, gdy wykonanie komendy nie powiedzie się.
- Jeśli komenda zakończy działanie z ostrzeżeniem, zostanie wysłana odpowiedź, jeśli zostanie ustawiona opcja MQRO_PAN lub MQRO_NAN.
- MQRO_PAN + MQRO_NAN oznacza, że menedżer kolejek wysyła odpowiedź bez względu na to, czy wykonanie komendy powiedzie się, czy nie. Ma to taki sam efekt z perspektywy menedżera kolejek, jak ustawienie parametru MsgType na wartość MQMT_REQUEST.
- Jeśli nie ustawiono opcji MQRO_PAN ani MQRO_NAN, nie jest wysyłana żadna odpowiedź.

Formatowanie
Ustaw wartość MQFMT_RF_HEADER_2

MsgId
To pole jest zwykle ustawione na wartość MQMI_NONE, dzięki czemu menedżer kolejek generuje unikalną wartość.

CorrelId
W tym polu można ustawić dowolną wartość. Jeśli tożsamość nadawcy zawiera identyfikator CorrelId, należy podać tę wartość wraz z wartością MQRO_PASS_CORREL_ID w polu Raport , aby upewnić się, że jest ona ustawiona we wszystkich komunikatach odpowiedzi wysyłanych przez menedżer kolejek do nadawcy.

Kolejka_zwrotna
To pole definiuje kolejkę, do której mają być wysyłane odpowiedzi (jeśli istnieją). Może to być kolejka nadawcy. Jest to zaleta, że parametr QName może zostać pominięty w komunikacie. Jeśli jednak odpowiedzi mają być wysyłane do innej kolejki, wymagany jest parametr QName .

ReplyToQMgr
To pole definiuje menedżer kolejek dla odpowiedzi. Jeśli to pole pozostanie puste (wartość domyślna), menedżer kolejek lokalnych umieści w tym polu własną nazwę.

Ustawienia MQMD dla publikacji przekazywanych przez menedżer kolejek

Menedżer kolejek używa tych ustawień pól w deskryptorze komunikatu (MQMD) podczas wysyłania publikacji do subskrybenta. Wszystkie pozostałe pola w strukturze MQMD są ustawione na wartości domyślne.

Raport
Opcja Raport jest ustawiona na wartość MQRO_NONE.

MsgType
MsgType ma ustawioną wartość MQMT_DATAGRAM.

Utrata ważności
Utrata ważności jest ustawiana na wartość podaną w komunikacie Publikowanie odebranym od publikatora. W przypadku zachowanego komunikatu czas oczekiwania jest zmniejszany o przybliżony czas przebyowania komunikatu w menedżerze kolejek.

Formatowanie
Format jest ustawiony na wartość MQFMT_RF_HEADER_2

MsgId
MsgId ma ustawioną unikalną wartość.

CorrelId
Jeśli identyfikator CorrelId jest częścią tożsamości subskrybenta, jest to wartość określona przez subskrybenta podczas rejestrowania. W przeciwnym razie jest to wartość niezerowa wybrana przez menedżer kolejek.

Priorytet
Priorytet przyjmuje wartość ustawioną przez publikator lub jako rozstrzygniętą, jeśli publikator określił wartość MQPRI_PRIORITY_AS_Q_DEF.

Trwałość
Trwałość przyjmuje wartość ustawioną przez publikator lub jako rozstrzygniętą, jeśli publikator określił wartość MQPER_PERSISTENCE_AS_Q_DEF, chyba że określono inaczej w komunikacie Zarejestruj subskrybenta dla subskrybenta, do którego ta publikacja jest wysyłana.

Kolejka_zwrotna
ReplyToQ jest puste.

ReplyToQMgr
Parametr ReplyToQMgr jest ustawiony na nazwę menedżera kolejek.

UserIdentifier
UserIdentifier jest identyfikatorem użytkownika subskrybenta ustawionym podczas rejestrowania subskrybenta.

AccountingToken
AccountingToken jest tokenem rozliczeniowym subskrybenta ustawionym podczas pierwszej rejestracji subskrybenta.

Dane_tożsamości_aplikacji
ApplIdentityDane to dane tożsamości aplikacji subskrybenta ustawione podczas pierwszej rejestracji subskrybenta.

PutApplType,
PutApplTyp jest ustawiony na wartość MQAT_BROKER.

PutApplName,
Parametr PutApplNazwa jest ustawiony na pierwszych 28 znaków nazwy menedżera kolejek.

PutDate
PutDate jest datą umieszczenia komunikatu.

PutTime
PutTime to czas umieszczenia komunikatu.

ApplOriginData.
Pole ApplOriginData jest puste.

Ustawienia MQMD w komunikatach odpowiedzi menedżera kolejek

Menedżer kolejek używa tych ustawień pól w deskryptorze komunikatu (MQMD) podczas wysyłania odpowiedzi na komunikat publikacji. Wszystkie pozostałe pola w strukturze MQMD są ustawione na wartości domyślne.

Raport
W polu Raport są ustawione same zera.

MsgType
Parametr MsgType jest ustawiony na wartość MQMT_REPLY.

Formatowanie
Format jest ustawiony na wartość MQFMT_RF_HEADER_2

MsgId
Ustawienie pola MsgId jest zależne od opcji Report w oryginalnym komunikacie komendy. Domyślnie jest ona ustawiona na wartość MQMI_NONE, dzięki czemu menedżer kolejek generuje unikalną wartość.

CorrelId
Ustawienie CorrelId zależy od opcji Report (Raport) w oryginalnym komunikacie komendy. Domyślnie oznacza to, że parametr CorrelId jest ustawiony na tę samą wartość, co parametr MsgId komunikatu komendy. Można go użyć do korelowania komend z ich odpowiedziami.

Priorytet
Priorytet jest ustawiany na taką samą wartość, jak w oryginalnym komunikacie komendy.

Trwałość
Trwałość jest ustawiona na wartość ustawioną w oryginalnym komunikacie komendy.

Utrata ważności
Utrata ważności jest ustawiana na taką samą wartość, jak w oryginalnym komunikacie komendy odebranym przez menedżer kolejek.

PutApplType,
PutApplTyp jest ustawiony na wartość MQAT_BROKER.

PutApplName,
Parametr PutApplNazwa jest ustawiony na pierwsze 28 znaków nazwy menedżera kolejek.

Inne pola kontekstu są ustawiane tak, jakby były generowane za pomocą opcji MQPMO_PASS_IDENTITY_CONTEXT.

Kodowania maszynowe

W tej sekcji opisano strukturę pola *Encoding* w deskryptorze komunikatu.

Podsumowanie pól w strukturze zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432 .

Pole *Encoding* jest 32-bitową liczbą całkowitą podzieloną na cztery osobne podpola. Te podpola identyfikują:

- Kodowanie używane dla binarnych liczb całkowitych
- Kodowanie używane dla liczb całkowitych z upakowaną liczbą dziesiętną
- Kodowanie używane dla liczb zmiennopozycyjnych
- Zarezerwowane bity

Każde podpole jest identyfikowane przez maskę bitową, która ma 1 bity w pozycjach odpowiadających podpolu i 0 bitów w innym miejscu. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 najmniej znaczącym bitem. Zdefiniowane są następujące maski:

MQENC_INTEGER_MASK (maska_liczba_całk)
Maska dla kodowania binarno-całkowitoliczbowe.

To podpole zajmuje pozycje bitów od 28 do 31 w polu *Encoding* .

MQENC_DECIMAL_MASK
Maska kodowania upakowanego dziesiętnego typu integer.

To podpole zajmuje pozycje bitów od 24 do 27 w polu *Encoding* .

MQENC_FLOAT_MASK
Maska kodowania zmiennopozycyjnego.

To podpole zajmuje pozycje bitów od 20 do 23 w polu *Encoding* .

MASKA_ZAREZERWOWANA_MQENC_
Maska zarezerwowanych bitów.

To podpole zajmuje pozycje bitów od 0 do 19 w polu *Encoding* .

Kodowanie binarne-liczba całkowita

Następujące wartości są poprawne dla kodowania binarna-liczba całkowita:

MQENC_INTEGER_UNDEFINED,
Binarne liczby całkowite są reprezentowane przy użyciu niezdefiniowanego kodowania.

MQENC_INTEGER_NORMAL
Binarne liczby całkowite są reprezentowane w konwencjonalny sposób:

- Najmniej znaczący bajt w liczbie ma najwyższy adres spośród wszystkich bajtów w liczbie; najmniej znaczący bajt ma najniższy adres
- Najmniej znaczący bit w każdym bajcie przylega do bajtu z następnym wyższym adresem; najbardziej znaczący bit w każdym bajcie przylega do bajtu z następnym niższym adresem

MQENC_INTEGER_REVERSED,
Binarne liczby całkowite są reprezentowane w taki sam sposób, jak MQENC_INTEGER_NORMAL, ale z bajtami ułożonymi w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak MQENC_INTEGER_NORMAL.

Kodowanie z upakowaną liczbą dziesiętną i liczbą całkowitą

Następujące wartości są poprawne dla kodowania liczb dziesiętnych upakowanych liczb całkowitych:

MQENC_DECIMAL_UNDEFINED,
Spakowane dziesiętne liczby całkowite są reprezentowane przy użyciu niezdefiniowanego kodowania.

MQENC_DECIMAL_NORMAL
Upakowane dziesiętne liczby całkowite są reprezentowane w konwencjonalny sposób:

- Każda cyfra dziesiętna w drukowalnej postaci liczby jest reprezentowana w upakowanej liczbie dziesiętnej przez pojedynczą cyfrę szesnastkową z zakresu od X' 0 'do X' 9'. Każda cyfra szesnastkowa zajmuje cztery bity, a każdy bajt w upakowanej liczbie dziesiętnej reprezentuje dwie cyfry dziesiętne w drukowalnej postaci liczby.
- Najmniej znaczący bajt w upakowanej liczbie dziesiętnej to bajt, który zawiera najmniej znaczącą cyfrę dziesiętną. W obrębie tego bajtu najbardziej znaczące cztery bity zawierają najmniej znaczącą cyfrę dziesiętną, a najmniej znaczące cztery bity zawierają znak. Znakiem jest X'C '(dodatni), X 'D' (ujemny) lub X'F ' (niepodpisany).
- Najmniej znaczący bajt w liczbie ma najwyższy adres spośród wszystkich bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres.
- Najmniej znaczący bit w każdym bajcie przylega do bajtu z następnym wyższym adresem; najbardziej znaczący bit w każdym bajcie przylega do bajtu z następnym niższym adresem.

MQENC_DECIMAL_REVERSED (odwrócone)
Upakowane dziesiętne liczby całkowite są reprezentowane w taki sam sposób, jak MQENC_DECIMAL_NORMAL, ale z bajtami ułożonymi w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak MQENC_DECIMAL_NORMAL.

Kodowanie zmiennopozycyjne

Następujące wartości są poprawne dla kodowania zmiennopozycyjnego:

MQENC_FLOAT_UNDEFINED,
Liczby zmiennopozycyjne są reprezentowane przy użyciu niezdefiniowanego kodowania.

MQENC_FLOAT_IEEE_NORMAL
Liczby zmiennopozycyjne są reprezentowane przy użyciu standardowego IEEE[4]Format zmiennopozycyjny, w którym bajty są ułożone w następujący sposób:

[4] Instytut Inżynierów Elektroniki i Elektroniki

- Najmniej znaczący bajt w mantysie ma najwyższy adres spośród wszystkich bajtów w liczbie; bajt zawierający wykładnik ma najniższy adres
- Najmniej znaczący bit w każdym bajcie przylega do bajtu z następnym wyższym adresem; najbardziej znaczący bit w każdym bajcie przylega do bajtu z następnym niższym adresem

Szczegóły kodowania zmiennopozycyjnego IEEE można znaleźć w standardzie IEEE 754.

MQENC_FLOAT_IEEE_REVERSED
Liczby zmiennopozycyjne są reprezentowane w taki sam sposób, jak MQENC_FLOAT_IEEE_NORMAL, ale z bajtami ułożonymi w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak MQENC_FLOAT_IEEE_NORMAL.

MQENC_FLOAT_S390
Liczby zmiennopozycyjne są reprezentowane przez standardowy format zmiennopozycyjny System/390 ; jest on również używany przez system System/370.

Konstruowanie kodowania

Aby utworzyć wartość dla pola *Encoding* w strukturze MQMD, można dodać odpowiednie stałe opisujące wymagane kodowania (nie należy dodawać tej samej stałej więcej niż raz) lub połączyć je za pomocą bitowej operacji OR (jeśli język programowania obsługuje operacje bitowe).

Niezależnie od użytej metody należy łączyć tylko jedno z kodowań MQENC_INTEGER_ * z jednym z kodowań MQENC_DECIMAL_ * i jednym z kodowań MQENC_FLOAT_ *.

Analizowanie kodowania

Pole *Encoding* zawiera podpola. W związku z tym aplikacje, które muszą sprawdzić kodowanie liczb całkowitych, upakowanych liczb dziesiętnych lub zmiennopozycyjnych, muszą używać jednej z opisanych technik.

Używanie operacji bitowych

Jeśli język programowania obsługuje operacje bitowe, wykonaj następujące kroki:

1. Wybierz jedną z następujących wartości, w zależności od wymaganego typu kodowania:
   - MQENC_INTEGER_MASK dla binarnego kodowania liczb całkowitych
   - MQENC_DECIMAL_MASK dla kodowania liczb dziesiętnych upakowanych
   - MQENC_FLOAT_MASK dla kodowania zmiennopozycyjnego

   Wywołaj wartość A.
2. Połącz pole *Encoding* z wartością A , używając operacji bitowej AND; wywołaj wynik B.
3. B jest wymaganym kodowaniem i można go przetestować pod kątem równości z każdą wartością, która jest poprawna dla tego typu kodowania.

Używanie arytmetyki

Jeśli język programowania *nie obsługuje* operacji bitowych, wykonaj następujące kroki, używając arytmetyki liczb całkowitych:

1. Wybierz jedną z następujących wartości, w zależności od wymaganego typu kodowania:
   - 1 dla binarnego kodowania liczb całkowitych
   - 16 dla kodowania liczb całkowitych z upakowanym kodem dziesiętnym
   - 256 dla kodowania zmiennopozycyjnego

   Wywołaj wartość A.
2. Podziel wartość pola *Encoding* przez A . Wywołaj wynik B.
3. Podziel B przez 16; wywołaj wynik C.

4. Pomnóż C przez 16 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. E jest wymaganym kodowaniem i można go przetestować pod kątem równości z każdą wartością, która jest poprawna dla tego typu kodowania.

Podsumowanie kodowania architektury maszynowej

Kodowania dla architektur komputerów przedstawia Tabela 631 na stronie 933.

Tabela 631. Podsumowanie kodowania dla architektur maszynowych

Architektura komputera	Binarne kodowanie całkowite	Kodowanie liczb dziesiętnych upakowanych liczb całkowitych	Kodowanie zmiennopozycyjne
IBM i	normalne	normalne	Normalny IEEE
Intel x86	Odwrotne	Odwrotne	Odwrócone IEEE
PowerPC	normalne	normalne	Normalny IEEE
System/390	normalne	normalne	System/390

Opcje raportu i flagi komunikatów

W tej sekcji opisano pola *Report* i *MsgFlags* , które są częścią deskryptora komunikatu MQMD określonego w wywołaniach MQGET, MQPUT i MQPUT1 .

Tematy w tej sekcji opisują:

- Struktura pola raportu i sposób jego przetwarzania przez menedżer kolejek
- W jaki sposób aplikacja analizuje pole raportu
- Struktura pola message-flags

Więcej informacji na temat deskryptora komunikatu MQMD zawiera sekcja “MQMD-deskryptor komunikatu” na stronie 432.

Struktura pola raportu

Informacje te opisują strukturę pola raportu.

Pole *Report* jest 32-bitową liczbą całkowitą podzieloną na trzy osobne podpola. Te podpola identyfikują:

- Opcje raportu, które są odrzucane, jeśli menedżer kolejek lokalnych ich nie rozpoznaje
- Opcje raportu, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek ich nie rozpoznaje
- Opcje raportu, które są akceptowane tylko wtedy, gdy spełnione są inne warunki

Każde podpole jest identyfikowane przez maskę bitową, która ma 1 bity w pozycjach odpowiadających podpolu i 0 bitów w innym miejscu. Bity w podpolu nie muszą być sąsiadujące. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 najmniej znaczącym bitem. Następujące maski są zdefiniowane w celu identyfikacji pól podrzędnych:

MQRO_REJECT_UNSUP_MASK (maska_odrzucenia_mqr)
Ta maska identyfikuje pozycje bitowe w polu *Report* , w których opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, powodują niepowodzenie wywołania MQPUT lub MQPUT1 z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_REPORT_OPTIONS_ERROR.

To podpole zajmuje pozycje bitów 3 i od 11 do 13.

MQRO_ACCEPT_UNSUP_MASK
Ta maska identyfikuje pozycje bitowe w polu *Report* , w których opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, są jednak akceptowane w wywołaniach MQPUT lub MQPUT1 . W tym przypadku zwracany jest kod zakończenia MQCC_WARNING z kodem przyczyny MQRC_UNKNOWN_REPORT_OPTION.

To pole podrzędne zajmuje pozycje bitów od 0 do 2, od 4 do 10 i od 24 do 31.

W tym podpolu znajdują się następujące opcje raportu:

- DZIAŁANIE 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_UTR_WAŻN.
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA
- MQRO_NAN
- MQRO_NEW_MSG_ID
- MQRO_BRAK
- MQRO_PAN
- MQRO_PASS_CORREL_ID (Identyfikator KORELACJI MQ)
- MQRO_PASS_MSG_ID

MQRO_ACCEPT_UNSUP_IF_XMIT_MASK
Ta maska identyfikuje pozycje bitowe w polu *Report* , w których opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, są jednak akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem* , że spełnione są oba poniższe warunki:

- Komunikat jest przeznaczony dla zdalnego menedżera kolejek.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (czyli kolejka identyfikowana przez pola *ObjectQMgrName* i *ObjectName* w deskryptorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest lokalną kolejką transmisji).

Jeśli te warunki są spełnione, zwracany jest kod zakończenia MQCC_WARNING z kodem przyczyny MQRC_UNKNOWN_REPORT_OPTION, a jeśli nie, MQCC_FAILED z kodem przyczyny MQRC_REPORT_OPTIONS_ERROR.

To podpole zajmuje pozycje bitów od 14 do 23.

W tym podpolu znajdują się następujące opcje raportu:

- MQRO_COA
- MQRO_KOA_WITH_DATA
- MQRO_KOA_WITH_FULL_DATA
- MQRO_COD
- MQRO_KOD_WITH_DATA
- MQRO_KOD_WITH_FULL_DATA

Jeśli w polu *Report* są określone opcje, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza kolejno każde podpole, używając operacji bitowej AND w celu połączenia pola *Report* z maską dla tego podpola. Jeśli wynik tej operacji jest różny od zera, zwracany jest kod zakończenia i opisane wcześniej kody przyczyny.

Jeśli zostanie zwrócona wartość MQCC_WARNING, nie jest ona zdefiniowana, który kod przyczyny jest zwracany w przypadku wystąpienia innych warunków ostrzegawczych.

Możliwość określenia i zaakceptowania opcji raportu, które nie są rozpoznawane przez lokalny menedżer kolejek, jest przydatna podczas wysyłania komunikatu z opcją raportu rozpoznawaną i przetwarzaną przez *zdalny* menedżer kolejek.

Analizowanie pola raportu

Pole *Report* zawiera pola podrzędne; z tego powodu aplikacje, które muszą sprawdzić, czy nadawca komunikatu zażądał konkretnego raportu, muszą użyć jednej z opisanych technik.

Używanie operacji bitowych

Jeśli język programowania obsługuje operacje bitowe, wykonaj następujące kroki:

1. Wybierz jedną z następujących wartości, w zależności od typu raportu, który ma zostać sprawdzony:
   - MQRO_COA_WITH_FULL_DATA dla raportu COA
   - MQRO_COD_WITH_FULL_DATA dla raportu COD
   - MQRO_EXCEPTION_WITH_FULL_DATA dla raportu o wyjątku
   - MQRO_EXPIRATION_WITH_FULL_DATA dla raportu utraty ważności

   Wywołaj wartość A.

   W systemie z/OSnależy użyć wartości MQRO_ * _WITH_DATA zamiast wartości MQRO_ * _WITH_FULL_DATA.
2. Połącz pole *Report* z wartością A , używając operacji bitowej AND; wywołaj wynik B.
3. Przetestuj właściwość B pod kątem równości z każdą wartością, która jest możliwa dla tego typu raportu.

   Jeśli na przykład parametr A ma wartość MQRO_EXCEPTION_WITH_FULL_DATA, należy przetestować właściwość B pod kątem równości z każdym z poniższych kryteriów, aby określić, co zostało określone przez nadawcę komunikatu:

   - MQRO_BRAK
   - MQRO_EXCEPTION,
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Testy mogą być wykonywane w dowolnej kolejności, która jest najwygodniejsza dla logiki aplikacji.

Użyj podobnej metody, aby przetestować opcje MQRO_PASS_MSG_ID lub MQRO_PASS_CORREL_ID; wybierz wartość A , która z tych dwóch stałych jest odpowiednia, a następnie postępuj zgodnie z wcześniejszym opisem.

Używanie arytmetyki

Jeśli język programowania *nie obsługuje* operacji bitowych, wykonaj następujące kroki, używając arytmetyki liczb całkowitych:

1. Wybierz jedną z następujących wartości, w zależności od typu raportu, który ma zostać sprawdzony:
   - MQRO_COA dla raportu COA
   - MQRO_COD dla raportu COD
   - Wyjątek MQRO_EXCEPTION dla raportu o wyjątkach
   - MQRO_EXPIRATION dla raportu utraty ważności

   Wywołaj wartość A.
2. Podziel pole *Report* przez A . Wywołaj wynik B.

3. Podziel B przez 8 ; Wywołaj wynik C.
4. Pomnóż C przez 8 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. Przetestuj właściwość E pod kątem równości z każdą wartością, która jest możliwa dla tego typu raportu.

   Jeśli na przykład parametr A ma wartość MQRO_EXCEPTION, należy przetestować właściwość E pod kątem równości z każdym z poniższych kryteriów, aby określić, co zostało określone przez nadawcę komunikatu:

   - MQRO_BRAK
   - MQRO_EXCEPTION,
   - MQRO_EXCEPTION_WITH_DATA
   - MQRO_EXCEPTION_WITH_FULL_DATA

   Testy mogą być wykonywane w dowolnej kolejności, która jest najwygodniejsza dla logiki aplikacji.

Poniższa pseudocode ilustruje tę technikę dla komunikatów raportu o wyjątkach:

```
A = MQRO_EXCEPTION
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Użyj podobnej metody, aby przetestować opcje MQRO_PASS_MSG_ID lub MQRO_PASS_CORREL_ID; wybierz wartość A , która z tych dwóch stałych jest odpowiednia, a następnie wykonaj czynności opisane wcześniej, ale zastąp wartość 8 w poprzednich krokach wartością 2.

Struktura pola flagi komunikatu

Ta informacja opisuje strukturę pola flagi komunikatu.

Pole *MsgFlags* jest 32-bitową liczbą całkowitą podzieloną na trzy osobne podpola. Te podpola identyfikują:

- Flagi komunikatów, które są odrzucane, jeśli lokalny menedżer kolejek ich nie rozpoznaje
- Flagi komunikatów, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek ich nie rozpoznaje
- Flagi komunikatów, które są akceptowane tylko wtedy, gdy spełnione są inne warunki

Uwaga: Wszystkie podpola w programie *MsgFlags* są zarezerwowane do użytku przez menedżer kolejek.

Każde podpole jest identyfikowane przez maskę bitową, która ma 1 bity w pozycjach odpowiadających podpolu i 0 bitów w innym miejscu. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 najmniej znaczącym bitem. Następujące maski są zdefiniowane w celu identyfikacji pól podrzędnych:

MQMF_REJECT_UNSUP_MASK
Ta maska identyfikuje pozycje bitowe w polu *MsgFlags* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, powodują niepowodzenie wywołania MQPUT lub MQPUT1 z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_MSG_FLAGS_ERROR.

To podpole zajmuje pozycje bitów od 20 do 31.

W tym polu podrzędnym znajdują się następujące flagi komunikatów:

- MQMF_LAST_MSG_IN_GROUP
- MQMF_LAST_SEGMENT
- MQMF_MSG_IN_GROUP
- MQMF_SEGMENT

- MQMF_SEGMENTATION_ALLOWED
- MQMF_SEGMENTATION_INHIBITED

MQMF_ACCEPT_UNSUP_MASK
Ta maska identyfikuje pozycje bitowe w polu *MsgFlags* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, są mimo to akceptowane w wywołaniach MQPUT lub MQPUT1 . Kod zakończenia to MQCC_OK.

To podpole zajmuje pozycje bitów od 0 do 11.

MQMF_ACCEPT_UNSUP_IF_XMIT_MASK
Ta maska identyfikuje pozycje bitowe w polu *MsgFlags* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, są mimo to akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem* , że spełnione są oba następujące warunki:

- Komunikat jest przeznaczony dla zdalnego menedżera kolejek.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (czyli kolejka identyfikowana przez pola *ObjectQMgrName* i *ObjectName* w deskryptorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest lokalną kolejką transmisji).

Jeśli te warunki są spełnione, zwracany jest kod zakończenia MQCC_OK, a jeśli nie, MQCC_FAILED z kodem przyczyny MQRC_MSG_FLAGS_ERROR.

To podpole zajmuje pozycje bitów od 12 do 19.

Jeśli w polu *MsgFlags* określono flagi, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza kolejno każde podpole, używając operacji bitowej AND w celu połączenia pola *MsgFlags* z maską dla tego podpola. Jeśli wynik tej operacji jest różny od zera, zwracany jest kod zakończenia i opisane wcześniej kody przyczyny.

Wyjście konwersji danych

W tej kolekcji tematów opisano interfejs do wyjścia konwersji danych oraz przetwarzanie wykonywane przez menedżer kolejek, gdy wymagana jest konwersja danych.

Więcej informacji na temat konwersji danych zawiera sekcja *Konwersja danych w sekcji IBM MQ* pod adresem https://www.ibm.com/support/pages/node/317869.

Wyjście konwersji danych jest wywoływane w ramach przetwarzania wywołania MQGET w celu przekształcenia danych komunikatu aplikacji w reprezentację wymaganą przez aplikację odbierającą. Konwersja danych komunikatu aplikacji jest opcjonalna. Wymaga ona określenia opcji MQGMO_CONVERT w wywołaniu MQGET.

Opisane są następujące tematy:

- Przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję MQGMO_CONVERT. Patrz sekcja “Przetwarzanie konwersji” na stronie 938.
- Konwencje przetwarzania używane przez menedżer kolejek podczas przetwarzania wbudowanego formatu. Konwencje te są zalecane również w przypadku programów zewnętrznych napisanych przez użytkownika. Patrz “Konwencje przetwarzania” na stronie 939.
- Specjalne uwagi dotyczące przekształcania komunikatów raportu; patrz sekcja “Konwersja komunikatów raportu” na stronie 943.
- Parametry przekazywane do wyjścia konwersji danych; patrz sekcja “MQ_DATA_CONV_EXIT-wyjście konwersji danych” na stronie 956.
- Wywołanie, którego można użyć z wyjścia do konwersji danych znakowych między różnymi reprezentacjami; patrz “MQXCNVC-przekształcanie znaków” na stronie 950.
- Parametr struktury danych specyficzny dla wyjścia; patrz sekcja “MQDXP-parametr wyjścia konwersji danych” na stronie 944.

Przetwarzanie konwersji

Te informacje opisują przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję MQGMO_CONVERT.

Menedżer kolejek wykonuje następujące działania, jeśli w wywołaniu MQGET podano opcję MQGMO_CONVERT i istnieje komunikat, który ma zostać zwrócony do aplikacji:

1. Jeśli spełniony jest co najmniej jeden z poniższych warunków, konwersja nie jest konieczna:
   - Dane komunikatu są już w zestawie znaków i kodowaniu wymaganym przez aplikację wywołującą wywołanie MQGET. Przed wykonaniem wywołania aplikacja musi ustawić pola *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** wywołania MQGET na wymagane wartości.
   - Długość danych komunikatu wynosi zero.
   - Długość parametru **Buffer** wywołania MQGET wynosi zero.

   W takich przypadkach komunikat jest zwracany bez konwersji do aplikacji, która wywołała wywołanie MQGET. Wartości *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** są ustawiane na wartości w informacjach sterujących komunikatu, a wywołanie kończy się jedną z następujących kombinacji kodu zakończenia i kodu przyczyny:

Tabela 632. Kombinacje kodu zakończenia i kodu przyczyny

Kod zakończenia	Kod przyczyny
MQCC_OK	MQRC_BRAK
Ostrzeżenie MQCC	MQRC_TRUNCATED_MSG_ACCEPTED
Ostrzeżenie MQCC	Niepowodzenie MQRC_TRUNCATED_MSG_FAILED

   Poniższe kroki są wykonywane tylko wtedy, gdy zestaw znaków lub kodowanie danych komunikatu różni się od odpowiedniej wartości parametru **MsgDesc** i istnieją dane do przekształcenia:

2. Jeśli pole *Format* w informacjach sterujących komunikatu ma wartość MQFMT_NONE, komunikat jest zwracany bez konwersji z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_FORMAT_ERROR.

   We wszystkich innych przypadkach przetwarzanie konwersji jest kontynuowane.

3. Komunikat zostanie usunięty z kolejki i umieszczony w tymczasowym buforze, którego wielkość jest taka sama jak wielkość parametru **Buffer** . W przypadku operacji przeglądania komunikat jest kopiowany do buforu tymczasowego, a nie usuwany z kolejki.
4. Jeśli komunikat musi zostać obcięty, aby zmieścił się w buforze, wykonywane są następujące czynności:
   - Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie została określona, komunikat jest zwracany bez konwersji z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_FAILED.
   - Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG *została* określona, kod zakończenia jest ustawiany na wartość MQCC_WARNING, kod przyczyny jest ustawiany na wartość MQRC_TRUNCATED_MSG_ACCEPTED, a przetwarzanie konwersji jest kontynuowane.
5. Jeśli komunikat może zostać zapisany w buforze bez obcięcia lub podano opcję MQGMO_ACCEPT_TRUNCATED_MSG, wykonywane są następujące czynności:
   - Jeśli format jest wbudowany, bufor jest przekazywany do usługi konwersji danych menedżera kolejek.
   - Jeśli format nie jest formatem wbudowanym, bufor jest przekazywany do wyjścia napisanego przez użytkownika o takiej samej nazwie jak format. Jeśli nie można znaleźć wyjścia, komunikat jest zwracany bez konwersji z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_FORMAT_ERROR.

Jeśli nie wystąpi żaden błąd, dane wyjściowe z usługi konwersji danych lub z wyjścia napisanego przez użytkownika są przekształcanym komunikatem oraz kodem zakończenia i kodem przyczyny, który ma zostać zwrócony do aplikacji wywołującej wywołanie MQGET.

6. Jeśli konwersja powiedzie się, menedżer kolejek zwraca przekształcony komunikat do aplikacji. W takim przypadku kod zakończenia i kod przyczyny zwracane przez wywołanie MQGET są jedną z następujących kombinacji:

Tabela 633. Kombinacje kodu zakończenia i kodu przyczyny

Kod zakończenia	**Kod przyczyny**
MQCC_OK	MQRC_BRAK
Ostrzeżenie MQCC	MQRC_TRUNCATED_MSG_ACCEPTED

Jeśli jednak konwersja jest wykonywana przez program zewnętrzny napisany przez użytkownika, mogą zostać zwrócone inne kody przyczyny, nawet jeśli konwersja zakończyła się pomyślnie.

Jeśli konwersja nie powiedzie się, menedżer kolejek zwraca nieprzekształcony komunikat do aplikacji z polami *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** ustawionymi na wartości w informacjach sterujących komunikatu oraz z kodem zakończenia MQCC_WARNING.

Konwencje przetwarzania

Podczas przekształcania formatu wbudowanego menedżer kolejek jest zgodny z opisanymi konwencjami przetwarzania.

Procedury zewnętrzne napisane przez użytkownika również powinny być zgodne z tymi konwencjami, chociaż nie jest to wymuszane przez menedżer kolejek. Wbudowane formaty przekształcane przez menedżer kolejek to:

- MQFMT_ADMIN,
- MQFMT_CICS (tylko system z/OS)
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- Usługa MQFMT_DIST_HEADER
- MQFMT_EVENT, wersja 1
- MQFMT_EVENT, wersja 2
- MQFMT_IMS,
- MQFMT_IMS_VAR_STRING (łańcuch zmiennych)
- MQFMT_MD_EXTENSION
- MQFMT_PCF,
- MQFMT_REF_MSG_HEADER
- ZMQFMT_RF_HEADER
- MQFMT_RF_HEADER_2
- MQFMT_STRING
- MQFMT_TRIGGER,
- MQFMT_WORK_INFO_HEADER (tylko z/OS)
- MQFMT_XMIT_Q_HEADER

1. Jeśli komunikat jest rozwijany podczas konwersji i przekracza wielkość parametru **Buffer** , wykonywane są następujące czynności:

- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie została określona, komunikat jest zwracany bez konwersji z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_CONVERTED_MSG_TOO_BIG.
- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG *została* określona, komunikat został obcięty, kod zakończenia został ustawiony na wartość MQCC_WARNING, kod przyczyny został ustawiony na wartość MQRC_TRUNCATED_MSG_ACCEPTED, a przetwarzanie konwersji jest kontynuowane.

2. Jeśli wystąpi obcięcie (przed lub podczas konwersji), liczba poprawnych bajtów zwracanych w parametrze **Buffer** może być mniejsza niż długość buforu.

   Może to wystąpić na przykład wtedy, gdy 4-bajtowa liczba całkowita lub znak DBCS znajduje się na końcu buforu. Niekompletny element informacji nie jest przekształcany, a bajty w zwróconym komunikacie nie zawierają poprawnych informacji. Może to również wystąpić, jeśli komunikat, który został obcięty przed konwersją, zostanie zmniejszony podczas konwersji.

   Jeśli liczba zwróconych poprawnych bajtów jest mniejsza niż długość buforu, nieużywane bajty na końcu buforu są ustawiane na wartości null.
3. Jeśli tablica lub łańcuch znajduje się na końcu buforu, konwertowana jest jak najwięcej danych; nie jest konwertowany tylko konkretny element tablicy lub znak DBCS, który jest niekompletny; konwertowane są poprzedzające go elementy tablicy lub znaki.
4. Jeśli wystąpi obcięcie (przed lub podczas konwersji), długość zwracana dla parametru **DataLength** jest długością nieprzekształconego komunikatu przed obcięciem.
5. Jeśli łańcuchy są konwertowane między zestawami znaków jednobajtowych (SBCS), zestawami znaków dwubajtowych (DBCS) lub zestawami znaków wielobajtowych (MBCS), łańcuchy mogą się rozszerzać lub zwijać.
   - W formatach PCF MQFMT_ADMIN, MQFMT_EVENT i MQFMT_PCF łańcuchy w strukturach MQCFST i MQCFSL rozszerzają się lub kurczą, aby pomieścić łańcuch po konwersji.

     W przypadku struktury listy łańcuchów MQCFSL łańcuchy na liście mogą być rozwijane lub zwiane o różne kwoty. W takim przypadku menedżer kolejek dopełni krótsze łańcuchy odstępami, aby były one takie same jak najdłuższy łańcuch po konwersji.
   - W formacie MQFMT_REF_MSG_HEADER łańcuchy zaadresowane w polach `SrcEnvOffset`, `SrcNameOffset`, `DestEnvOffset`i `DestNameOffset` są odpowiednio rozwijane lub zwiane, aby pomieścić łańcuchy po konwersji.
   - W formacie MQFMT_RF_HEADER pole `NameValueString` jest rozszerzane lub zwijane zgodnie z potrzebami w celu uwzględnienia par nazwa-wartość po konwersji.
   - W strukturach o stałej wielkości pól menedżer kolejek zezwala na rozwijanie lub zwijania łańcuchów w obrębie ich stałych pól, pod warunkiem, że nie zostaną utracone żadne istotne informacje. W związku z tym końcowe odstępy i znaki występujące po pierwszym znaku o kodzie zero w polu są traktowane jako nieistotne.
     - Jeśli łańcuch zostanie rozwinięty, ale tylko nieistotne znaki muszą zostać usunięte, aby pomieścić przekształcony łańcuch w polu, konwersja powiedzie się, a wywołanie zostanie zakończone z błędem MQCC_OK i kodem przyczyny MQRC_NONE (przy założeniu braku innych błędów).
     - Jeśli łańcuch zostanie rozwinięty, ale przekształcony łańcuch wymaga usunięcia znaczących znaków, aby zmieścił się w polu, komunikat jest zwracany bez konwersji, a wywołanie kończy się komunikatem MQCC_WARNING i kodem przyczyny MQRC_CONVERTED_STRING_TOO_BIG.

       Uwaga: Kod przyczyny MQRC_CONVERTED_STRING_TOO_BIG powoduje w tym przypadku, czy określono opcję MQGMO_ACCEPT_TRUNCATED_MSG.
     - Jeśli łańcuch się kontrakty, menedżer kolejek dopełnić łańcuch odstępami do długości pola.
6. W przypadku komunikatów składających się z co najmniej jednej struktury nagłówka produktu MQ , po której następują dane użytkownika, może zostać przekształcona co najmniej jedna struktura nagłówka, a pozostała część komunikatu nie. Jednak (z dwoma wyjątkami) pola *CodedCharSetId* i *Encoding* w każdej strukturze nagłówka zawsze poprawnie wskazują zestaw znaków i kodowanie danych, które następują po strukturze nagłówka.

Dwa wyjątki to struktury MQCIH i MQIIH, w których wartości w polach *CodedCharSetId* i *Encoding* w tych strukturach nie są istotne. W przypadku tych struktur dane znajdujące się po strukturze mają ten sam zestaw znaków i kodowanie, co struktura MQCIH lub MQIIH.

7. Jeśli pola *CodedCharSetId* lub *Encoding* w informacjach sterujących pobieranego komunikatu lub w parametrze **MsgDesc** określają wartości, które nie są zdefiniowane lub nie są obsługiwane, menedżer kolejek może zignorować błąd, jeśli niezdefiniowana lub nieobsługiwana wartość nie musi być używana podczas przekształcania komunikatu.

   Jeśli na przykład pole *Encoding* w komunikacie określa nieobsługiwane kodowanie zmiennopozycyjne, ale komunikat zawiera tylko dane całkowite lub zawiera dane zmiennopozycyjne, które nie wymagają konwersji (ponieważ kodowanie źródłowe i docelowe są identyczne), błąd może nie zostać rozpoznany.

   W przypadku zdiagnozowania błędu komunikat jest zwracany bez konwersji z kodem zakończenia MQCC_WARNING i jednym z kodów przyczyny MQRC_SOURCE_ * _ERROR lub MQRC_TARGET_ * _ERROR (odpowiednio); pola *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** są ustawiane na wartości w informacjach sterujących komunikatu.

   Jeśli błąd nie zostanie rozpoznany, a konwersja zakończy się pomyślnie, wartości zwracane w polach *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** są wartościami określonymi przez aplikację wywołującą wywołanie MQGET.

8. We wszystkich przypadkach, jeśli komunikat jest zwracany do aplikacji, która nie została przekształcona, kod zakończenia jest ustawiany na wartość MQCC_WARNING, a pola *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** są ustawiane na wartości odpowiednie dla nieprzekształconych danych. Jest to również wykonywane dla MQFMT_NONE.

   Parametr **Reason** jest ustawiony na kod, który wskazuje, dlaczego nie można przeprowadzić konwersji, chyba że komunikat również musiał zostać obcięty. Kody przyczyny związane z obcięciem mają pierwszeństwo przed kodami przyczyny związanymi z konwersją. (Aby określić, czy obcięty komunikat został przekształcony, sprawdź wartości zwrócone w polach *CodedCharSetId* i *Encoding* w parametrze **MsgDesc**).

   Po zdiagnozowaniu błędu zwracany jest konkretny kod przyczyny lub ogólny kod przyczyny MQRC_NOT_CONVERTED. Zwrócony kod przyczyny zależy od możliwości diagnostycznych bazowej usługi konwersji danych.

9. Jeśli zostanie zwrócony kod zakończenia MQCC_WARNING i ma zastosowanie więcej niż jeden kod przyczyny, kolejność wykonywania jest następująca:

   a. Następujące przyczyny mają pierwszeństwo przed wszystkimi innymi. Może wystąpić tylko jedna z przyczyn w tej grupie:

      - MQRC_SIGNAL_REQUEST_ACCEPTED
      - MQRC_TRUNCATED_MSG_ACCEPTED

   b. Kolejność wykonywania operacji w pozostałych kodach przyczyny nie jest zdefiniowana.

10. Po zakończeniu wywołania MQGET:

    - Następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:
      - MQRC_BRAK
    - Następujące kody przyczyny wskazują, że komunikat *mógł* zostać pomyślnie przekształcony (sprawdź pola *CodedCharSetId* i *Encoding* w parametrze **MsgDesc** , aby uzyskać więcej informacji):
      - MQRC_MSG_MARKED_BROWSE_CO_OP
      - MQRC_TRUNCATED_MSG_ACCEPTED
    - Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

    Następujące przetwarzanie jest specyficzne dla formatów wbudowanych; nie dotyczy formatów zdefiniowanych przez użytkownika:

11. Z wyjątkiem następujących formatów:

- MQFMT_ADMIN,
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_EVENT, ZDARZENIE
- MQFMT_IMS_VAR_STRING (łańcuch zmiennych)
- MQFMT_PCF,
- MQFMT_STRING

żaden z wbudowanych formatów nie może być przekształcony z lub na zestawy znaków, które nie zawierają znaków SBCS dla znaków, które są poprawne w nazwach kolejek. Jeśli zostanie podjęta próba wykonania takiej konwersji, komunikat zostanie zwrócony bez konwersji, z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_SOURCE_CCSID_ERROR lub MQRC_TARGET_CCSID_ERROR (odpowiednio).

Zestaw znaków Unicode UTF-16 jest przykładem zestawu znaków, który nie zawiera znaków SBCS dla znaków, które są poprawne w nazwach kolejek.

12. Jeśli dane komunikatu dla wbudowanego formatu są obcięte, pola w komunikacie, które zawierają długości łańcuchów lub liczby elementów lub struktur, nie są dopasowywane w celu odzwierciedlenia długości danych rzeczywiście zwróconych do aplikacji. Wartości zwracane dla takich pól w danych komunikatu są wartościami mającymi zastosowanie do komunikatu *przed obcięciem*.

    Podczas przetwarzania komunikatów, takich jak obcięty komunikat MQFMT_ADMIN, należy upewnić się, że aplikacja nie próbuje uzyskać dostępu do danych poza końcem zwracanych danych.

13. Jeśli nazwa formatu to MQFMT_DEAD_LETTER_HEADER, dane komunikatu rozpoczynają się od struktury MQDLH, po której może wystąpić zero lub więcej bajtów danych komunikatu aplikacji. Format, zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane przez pola `Format`, `CodedCharSetId`i `Encoding` w strukturze MQDLH na początku komunikatu. Ponieważ struktura MQDLH i dane komunikatu aplikacji mogą mieć różne zestawy znaków i kodowania, jedna ze struktur MQDLH i dane komunikatu aplikacji mogą wymagać konwersji.

    W razie potrzeby menedżer kolejek najpierw przekształca strukturę MQDLH. Jeśli konwersja powiedzie się lub struktura MQDLH nie wymaga konwersji, menedżer kolejek sprawdza pola `CodedCharSetId` i `Encoding` w strukturze MQDLH, aby sprawdzić, czy konwersja danych komunikatu aplikacji jest wymagana. Jeśli konwersja jest wymagana, menedżer kolejek wywołuje zapisane przez użytkownika wyjście z nazwą nadaną przez pole `Format` w strukturze MQDLH lub wykonuje samą konwersję (jeśli `Format` jest nazwą wbudowanego formatu).

    Jeśli wywołanie MQGET zwraca kod zakończenia MQCC_WARNING, a kod przyczyny wskazuje, że konwersja nie powiodła się, ma zastosowanie jedna z następujących sytuacji:

    - Nie można przekształcić struktury MQDLH. W takim przypadku dane komunikatu aplikacji nie zostaną również przekształcone.
    - Struktura MQDLH została przekształcona, ale dane komunikatu aplikacji nie zostały przekształcone.

    Aplikacja może sprawdzić wartości zwrócone w polach `CodedCharSetId` i `Encoding` w parametrze **`MsgDesc`** oraz w strukturze MQDLH w celu określenia, które z powyższych wartości mają zastosowanie.

14. Jeśli nazwa formatu to MQFMT_XMIT_Q_HEADER, dane komunikatu rozpoczynają się od struktury MQXQH, po której może wystąpić zero lub więcej bajtów dodatkowych danych. Te dodatkowe dane są zwykle danymi komunikatu aplikacji (mogą mieć zerową długość), ale na początku dodatkowych danych może być również obecna jedna lub więcej struktur nagłówka MQ .

    Struktura MQXQH musi być w zestawie znaków i kodowaniu menedżera kolejek. Format, zestaw znaków i kodowanie danych następujących po strukturze MQXQH są określone przez pola `Format`, `CodedCharSetId`i `Encoding` w strukturze MQMD zawartej w MQXQH. Dla każdej kolejnej obecnej struktury nagłówka produktu MQ pola `Format`, `CodedCharSetId`i `Encoding` w strukturze opisują dane, które następują po tej strukturze. Dane te są albo inną strukturą nagłówka produktu MQ , albo danymi komunikatu aplikacji.

Jeśli opcja MQGMO_CONVERT jest określona dla komunikatu MQFMT_XMIT_Q_HEADER, dane komunikatu aplikacji i niektóre struktury nagłówka produktu MQ są przekształcane *, ale dane w strukturze MQXQH nie są*. W przypadku powrotu z wywołania MQGET:

- Wartości pól `Format`, `CodedCharSetId`i `Encoding` w parametrze **MsgDesc** opisują dane w strukturze MQXQH, a nie dane komunikatu aplikacji. Wartości nie są zatem takie same, jak wartości określone przez aplikację, która wywołała wywołanie MQGET.

  W rezultacie aplikacja, która wielokrotnie pobiera komunikaty z kolejki transmisji z określoną opcją MQGMO_CONVERT, musi zresetować pola `CodedCharSetId` i `Encoding` w parametrze **MsgDesc** do wartości wymaganych dla danych komunikatu aplikacji przed każdym wywołaniem MQGET.
- Wartości pól `Format`, `CodedCharSetId`i `Encoding` w ostatniej strukturze nagłówka produktu MQ opisują dane komunikatu aplikacji. Jeśli nie ma żadnych innych struktur nagłówka produktu MQ , dane komunikatu aplikacji są opisywane przez te pola w strukturze MQMD w strukturze MQXQH. Jeśli konwersja powiedzie się, wartości będą takie same jak wartości określone w parametrze **MsgDesc** przez aplikację, która wywołała wywołanie MQGET.

Jeśli komunikat jest komunikatem listy dystrybucyjnej, po strukturze MQXQH następuje struktura MQDH (plus jej tablice rekordów MQOR i MQPMR), po której może następować zero lub więcej struktur nagłówka MQ i zero lub więcej bajtów danych komunikatu aplikacji. Podobnie jak struktura MQXQH, struktura MQDH musi być w zestawie znaków i kodowaniu menedżera kolejek i nie jest przekształcana w wywołaniu MQGET, nawet jeśli podano opcję MQGMO_CONVERT.

Opisane wcześniej przetwarzanie struktur MQXQH i MQDH jest przeznaczone przede wszystkim do użycia przez agenty kanału komunikatów podczas pobierania komunikatów z kolejek transmisji.

Konwersja komunikatów raportu

Ogólnie komunikat raportu może zawierać różne ilości danych komunikatu aplikacji, w zależności od opcji raportu określonych przez nadawcę oryginalnego komunikatu. Jednak raport działań może zawierać dane, ale bez opcji raportu, w której stała zawiera * _WITH_DATA.

W szczególności komunikat raportu może zawierać:

1. Brak danych komunikatu aplikacji
2. Niektóre dane komunikatu aplikacji z oryginalnego komunikatu

   Dzieje się tak, gdy nadawca oryginalnego komunikatu określa MQRO_ * _WITH_DATA, a komunikat jest dłuższy niż 100 bajtów.
3. Wszystkie dane komunikatu aplikacji z oryginalnego komunikatu

   Dzieje się tak, gdy nadawca oryginalnego komunikatu określa MQRO_ * _WITH_FULL_DATA lub MQRO_ * _WITH_DATA, a komunikat ma długość 100 bajtów lub jest krótszy.

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, kopiuje nazwę formatu z oryginalnego komunikatu do pola *`Format`* w informacjach sterujących w komunikacie raportu. Nazwa formatu w komunikacie raportu może zatem sugerować długość danych, która jest inna niż długość rzeczywiście obecna w komunikacie raportu (przypadki 1 i 2 wcześniej).

Jeśli podczas pobierania komunikatu raportu określono opcję MQGMO_CONVERT:

- Dla poprzedniego przypadku 1, wyjście konwersji danych nie jest wywoływane (ponieważ komunikat raportu nie zawiera danych).
- Dla poprzedniego przypadku 3, nazwa formatu poprawnie implikuje długość danych komunikatu.
- Jednak w przypadku poprzedniego przypadku 2 wyjście konwersji danych jest wywoływane w celu przekształcenia komunikatu, który jest *krótszy* niż długość określona przez nazwę formatu.

  Ponadto kod przyczyny przekazany do wyjścia to zazwyczaj MQRC_NONE (oznacza to, że kod przyczyny nie wskazuje, że komunikat został obcięty). Dzieje się tak, ponieważ dane komunikatu zostały obcięte przez *nadawcę* komunikatu raportu, a nie przez menedżer kolejek odbiorcy w odpowiedzi na wywołanie MQGET.

Ze względu na te możliwości wyjście konwersji danych nie może używać nazwy formatu w celu określenia długości przekazywanych do niego danych. Zamiast tego wyjście musi sprawdzić długość podanych danych i być przygotowane do przekształcenia mniejszej ilości danych niż długość określona przez nazwę formatu. Jeśli dane mogą zostać pomyślnie przekształcone, kod zakończenia MQCC_OK i kod przyczyny MQRC_NONE muszą zostać zwrócone przez wyjście. Długość danych komunikatu do przekształcenia jest przekazywana do wyjścia jako parametr **InBufferLength** .

Interfejs programistyczny wrażliwy na produkt

MQDXP-parametr wyjścia konwersji danych

Struktura MQDXP jest parametrem, który menedżer kolejek przekazuje do wyjścia konwersji danych podczas wywoływania wyjścia w celu przekształcenia danych komunikatu w ramach przetwarzania wywołania MQGET. Szczegółowe informacje na temat wyjścia konwersji danych zawiera opis wywołania komendy MQ_DATA_CONV_EXIT.

Dane znakowe w MQDXP znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to nadawane przez atrybut menedżera kolejek produktu **CodedCharSetId** . Dane liczbowe w MQDXP są kodowane na komputerze rodzimym; jest to wartość określona przez parametr MQENC_NATIVE.

Tylko pola *DataLength*, *CompCode*, *Reason*i *ExitResponse* w MQDXP mogą być zmieniane przez wyjście; zmiany w innych polach są ignorowane. Pola *DataLength* nie można jednak zmienić, jeśli przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego.

Gdy sterowanie powraca do menedżera kolejek z wyjścia, menedżer kolejek sprawdza wartości zwrócone w MQDXP. Jeśli zwrócone wartości nie są poprawne, menedżer kolejek kontynuuje przetwarzanie tak, jakby wyjście zwróciło wartość MQXDR_CONVERSION_FAILED w produkcie *ExitResponse* . Jednak menedżer kolejek ignoruje wartości pól *CompCode* i *Reason* zwrócone przez wyjście w tym przypadku i używa zamiast nich wartości, które miały w *danych wejściowych* do wyjścia. Następujące wartości w MQDXP powodują takie przetwarzanie:

- Pole *ExitResponse* nie jest polem MQXDR_OK i nie jest polem MQXDR_CONVERSION_FAILED
- Pole *CompCode* jest inne niż MQCC_OK i nie MQCC_WARNING
- *DataLength* pole mniejsze niż zero lub *DataLength* pole zmienione, gdy przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego.

Poniższa tabela zawiera podsumowanie pól w strukturze.

Tabela 634. Pola w MQDXP

Pole	Opis	Temat
StrucId	Identyfikator struktury	StrucId
Version	Numer wersji struktury	Wersja
AppOptions	Opcje aplikacji	AppOptions
Encoding	Kodowanie liczbowe wymagane przez aplikację	Kodowanie
CodedCharSetId	Zestaw znaków wymagany przez aplikację	CodedCharSetId
DataLength	Długość danych komunikatu w bajtach	DataLength
CompCode	Kod zakończenia	CompCode
Reason	Kod przyczyny kwalifikujący się *CompCode*	Powód
ExitResponse	Odpowiedź z wyjścia	ExitResponse

Tabela 634. Pola w MQDXP (kontynuacja)		
Pole	**Opis**	**Temat**
Hconn	Uchwyt połączenia	Hconn (Hconn)
pEntryPoints	Adres struktury MQIEP	PunktypEntry

Pola

Struktura MQDXP zawiera następujące pola; pola są opisane w porządku alfabetycznym.

AppOptions
Typ: MQLONG

Jest to kopia pola *Options* struktury MQGMO określonej przez aplikację wywołującą wywołanie MQGET. Wyjście może wymagać sprawdzenia, czy podano opcję MQGMO_ACCEPT_TRUNCATED_MSG.

Jest to pole wejściowe do wyjścia.

CodedCharSetId
Typ: MQLONG

Jest to identyfikator kodowanego zestawu znaków dla zestawu znaków wymaganego przez aplikację, która wywołała wywołanie MQGET. Więcej szczegółów zawiera pole *CodedCharSetId* w strukturze MQMD. Jeśli aplikacja określa wartość specjalną MQCCSI_Q_MGR w wywołaniu MQGET, menedżer kolejek zmienia tę wartość na rzeczywisty identyfikator zestawu znaków używanego przez menedżer kolejek przed wywołaniem wyjścia.

Jeśli konwersja powiedzie się, wyjście musi skopiować to do pola *CodedCharSetId* w deskryptorze komunikatu.

Jest to pole wejściowe do wyjścia.

CompCode
Typ: MQLONG

Po wywołaniu wyjścia zawiera ono kod zakończenia, który jest zwracany do aplikacji, która wywołała wywołanie MQGET, jeśli wyjście nie wykonuje żadnego działania. Zawsze jest to komunikat MQCC_WARNING, ponieważ komunikat został obcięty lub komunikat wymaga konwersji, która nie została jeszcze wykonana.

W przypadku danych wyjściowych z wyjścia to pole zawiera kod zakończenia, który ma zostać zwrócony do aplikacji w parametrze **CompCode** wywołania MQGET. Poprawne są tylko wartości MQCC_OK i MQCC_WARNING. Opis pola *Reason* zawiera sugestie dotyczące sposobu, w jaki wyjście może ustawić to pole na wyjściu.

Jest to pole wejściowe/wyjściowe do wyjścia.

DataLength
Typ: MQLONG

Po wywołaniu wyjścia to pole zawiera oryginalną długość danych komunikatu aplikacji. Jeśli komunikat został obcięty, aby zmieścił się w buforze udostępnionym przez aplikację, wielkość komunikatu przekazanego do wyjścia jest *mniejsza* niż wartość *DataLength*. Wielkość komunikatu przekazanego do wyjścia jest zawsze określana przez parametr **InBufferLength** wyjścia, bez względu na ewentualne obcięcie.

Obcięcie jest wskazywane przez pole *Reason* zawierające wartość MQRC_TRUNCATED_MSG_ACCEPTED na wejściu do wyjścia.

Większość konwersji nie wymaga zmiany tej długości, ale w razie potrzeby może to zrobić wyjście. Wartość ustawiona przez wyjście jest zwracana do aplikacji w parametrze **DataLength** wywołania MQGET. Nie można jednak zmienić tej długości, jeśli przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego. Wynika to z faktu, że zmiana długości spowodowałaby, że przesunięcia późniejszych segmentów w komunikacie logicznym byłyby niepoprawne.

Należy zauważyć, że jeśli wyjście ma zmienić długość danych, należy pamiętać, że menedżer kolejek już zdecydował, czy dane komunikatu mieszczą się w buforze aplikacji, na podstawie długości *nieprzekształconych* danych. Ta decyzja określa, czy komunikat jest usuwany z kolejki (lub przenoszony jest kursor przeglądania dla żądania przeglądania) i nie ma na niego wpływu żadna zmiana długości danych spowodowana przez konwersję. Z tego powodu zaleca się, aby wyjścia konwersji nie powodowały zmiany długości danych komunikatu aplikacji.

Jeśli konwersja znaków oznacza zmianę długości, łańcuch może zostać przekształcony w inny łańcuch o takiej samej długości w bajtach, obcinając końcowe odstępy lub dopełniając odstępami, jeśli jest to konieczne.

Wyjście nie jest wywoływane, jeśli komunikat nie zawiera danych komunikatu aplikacji, dlatego wartość *DataLength* jest zawsze większa niż zero.

Jest to pole wejściowe/wyjściowe do wyjścia.

Encoding
Typ: MQLONG

Kodowanie liczbowe wymagane przez aplikację.

Jest to kodowanie liczbowe wymagane przez aplikację wydającą wywołanie MQGET. Więcej szczegółów zawiera pole *Encoding* w strukturze MQMD.

Jeśli konwersja powiedzie się, wyjście kopiuje to do pola *Encoding* w deskryptorze komunikatu.

Jest to pole wejściowe do wyjścia.

ExitOptions
Typ: MQLONG

Jest to pole zastrzeżone, a jego wartością jest `0`.

ExitResponse
Typ: MQLONG

Odpowiedź z wyjścia.Jest ona ustawiana przez wyjście, aby wskazać powodzenie lub brak konwersji. Musi to być jedna z następujących wartości:

MQXDR_OK
Konwersja powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca do aplikacji, która wywołała wywołanie MQGET, następujące dane:

- Wartość pola *CompCode* na wyjściu z wyjścia
- Wartość pola *Reason* na wyjściu z wyjścia
- Wartość pola *DataLength* na wyjściu z wyjścia
- Zawartość buforu wyjściowego wyjścia *OutBuffer*. Liczba zwróconych bajtów jest mniejszą z wartości parametru **OutBufferLength** wyjścia oraz wartość pola *DataLength* w danych wyjściowych wyjścia.

  Jeśli pola *Encoding* i *CodedCharSetId* w parametrze deskryptora komunikatu wyjścia mają *oba* niezmienione wartości, menedżer kolejek zwraca:
- Wartość pól *Encoding* i *CodedCharSetId* w strukturze MQDXP na *wejściu* do wyjścia.

  Jeśli jedno lub oba pola *Encoding* i *CodedCharSetId* w parametrze deskryptora komunikatu wyjścia zostały zmienione, menedżer kolejek zwraca:
- Wartości pól *Encoding* i *CodedCharSetId* w parametrze deskryptora komunikatu wyjścia w danych wyjściowych wyjścia

MQXDR_CONVERSION_FAILED.
Konwersja nie powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca do aplikacji, która wywołała wywołanie MQGET, następujące dane:

- Wartość pola *CompCode* na wyjściu z wyjścia
- Wartość pola *Reason* na wyjściu z wyjścia
- Wartość pola *DataLength* w polu *input* dla wyjścia.
- Zawartość buforu wejściowego wyjścia *InBuffer*. Liczba zwracanych bajtów jest określona przez parametr **InBufferLength** .

Jeśli wyjście zmieniło wartość *InBuffer*, wyniki są niezdefiniowane.

ExitResponse jest polem wyjściowym z wyjścia.

Hconn
Typ: MQHCONN

Jest to uchwyt połączenia, którego można użyć w wywołaniu MQXCNVC. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

pEntryPoints
Typ: PMQIEP

Adres struktury MQIEP, za pośrednictwem której można tworzyć wywołania MQI i DCI.

Reason
Typ: MQLONG

Kod przyczyny określający *CompCode*.

Po wywołaniu wyjścia zawiera ono kod przyczyny, który jest zwracany do aplikacji, która wywołała wywołanie MQGET, jeśli wyjście nie podejmuje żadnych działań. Możliwe wartości to MQRC_TRUNCATED_MSG_ACCEPTED, co oznacza, że komunikat został obcięty w celu dopasowania do buforu udostępnionego przez aplikację, oraz MQRC_NOT_CONVERTED, co oznacza, że komunikat wymaga konwersji, ale nie został jeszcze wykonane.

W przypadku danych wyjściowych z wyjścia to pole zawiera przyczynę, która ma zostać zwrócona do aplikacji w parametrze **Reason** wywołania MQGET. Zalecane jest wykonanie następujących czynności:

- Jeśli parametr *Reason* miał wartość MQRC_TRUNCATED_MSG_ACCEPTED na wejściu do wyjścia, pola *Reason* i *CompCode* nie mogą być zmieniane bez względu na to, czy konwersja powiedzie się, czy nie.

  Jeśli pole *CompCode* nie jest polem MQCC_OK, aplikacja pobierająca komunikat może zidentyfikować niepowodzenie konwersji, porównując zwrócone wartości *Encoding* i *CodedCharSetId* w deskryptorze komunikatu z żądanymi wartościami. Natomiast aplikacja nie może odróżnić obciętego komunikatu od komunikatu, który pasuje do buforu. Z tego powodu parametr MQRC_TRUNCATED_MSG_ACCEPTED musi zostać zwrócony w preferencjach do dowolnej przyczyny, która wskazuje niepowodzenie konwersji.
- Jeśli parametr *Reason* miał inną wartość na wejściu do wyjścia:
  - Jeśli konwersja powiedzie się, parametr *CompCode* musi być ustawiony na wartość MQCC_OK, a parametr *Reason* na wartość MQRC_NONE.
  - Jeśli konwersja nie powiedzie się lub komunikat zostanie rozwinięty i będzie musiał zostać obcięty, aby zmieścić się w buforze, parametr *CompCode* musi być ustawiony na wartość MQCC_WARNING (lub pozostawiony bez zmian), a parametr *Reason* na jedną z wymienionych wartości, aby wskazać rodzaj niepowodzenia.

    Uwaga: jeśli komunikat po konwersji jest zbyt duży dla buforu, musi zostać obcięty tylko wtedy, gdy aplikacja, która wywołała wywołanie MQGET, określiła opcję MQGMO_ACCEPT_TRUNCATED_MSG:

    - Jeśli określono tę opcję, zwracana jest przyczyna MQRC_TRUNCATED_MSG_ACCEPTED.
    - Jeśli ta opcja nie została określona, komunikat jest zwracany bez konwersji z kodem przyczyny MQRC_CONVERTED_MSG_TOO_BIG.

Wymienione kody przyczyny są zalecane do użycia przez wyjście w celu wskazania przyczyny niepowodzenia konwersji, ale wyjście może zwrócić inne wartości z zestawu kodów MQRC_ *, jeśli uzna to za stosowne. Ponadto zakres wartości od MQRC_APPL_FIRST do MQRC_APPL_LAST jest przydzielany do użycia przez wyjście w celu wskazania warunków, które mają być używane przez wyjście do komunikacji z aplikacją wywołującą wywołanie MQGET.

Uwaga: Jeśli nie można pomyślnie przekształcić komunikatu, wyjście musi zwrócić wartość MQXDR_CONVERSION_FAILED w polu *ExitResponse* , aby menedżer kolejek zwrócił nieprzekształcony komunikat. Dzieje się tak niezależnie od kodu przyczyny zwróconego w polu *Reason* .

MQRC_APPL_FIRST
(900, X'384 ') Najmniejsza wartość dla kodu przyczyny zdefiniowanego przez aplikację.

MQRC_APPL_LAST
(999, X'3E7') Najwyższa wartość dla kodu przyczyny zdefiniowanego przez aplikację.

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

MQRC_NOT_CONVERTED (nie skonwertowany)
(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

BŁĄD MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Niepoprawny identyfikator źródłowego kodowanego zestawu znaków.

BŁĄD MQRC_SOURCE_DECIMAL_ENC_ERROR
(2113, X'841 ') Nie rozpoznano kodowania dziesiętnego w komunikacie.

MQRC_SOURCE_FLOAT_ENC_ERROR.
(2114, X'842 ') Nierozpoznane kodowanie zmiennopozycyjne w komunikacie.

BŁĄD MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840 ') Nierozpoznane kodowanie źródłowe liczby całkowitej.

BŁĄD MQRC_TARGET_CCSID_ERROR
(2115, X'843 ') Niepoprawny identyfikator kodowanego zestawu znaków.

BŁĄD MQRC_TARGET_DECIMAL_ENC_ERROR
(2117, X'845 ') Nierozpoznane kodowanie Packed-decimal określone przez odbiornik.

MQRC_TARGET_FLOAT_ENC_ERROR
(2118, X'846 ') Nierozpoznane kodowanie zmiennopozycyjne określone przez odbiornik.

BŁĄD MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844 ') Nie rozpoznano kodowania docelowej liczby całkowitej.

MQRC_TRUNCATED_MSG_ACCEPTED
(2079, X'81F') Zwrócono obcięty komunikat (przetwarzanie zakończone).

Jest to pole wejściowe/wyjściowe do wyjścia.

StrucId
Typ: MQCHAR4

Identyfikator struktury.Wartość musi być następująca:

MQDXP_STRUC_ID
Identyfikator struktury parametru wyjścia konwersji danych.

Dla języka programowania C zdefiniowana jest również stała MQDXP_STRUC_ID_ARRAY; ma ona taką samą wartość jak MQDXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe do wyjścia.

Version
Typ: MQLONG

Numer wersji struktury.Wartość musi być następująca:

MQDXP_VERSION_1
Numer wersji struktury parametru wyjścia konwersji danych.

Następująca stała określa numer wersji bieżącej:

MQDXP_CURRENT_VERSION
Bieżąca wersja struktury parametru wyjścia konwersji danych.

Uwaga: Po wprowadzeniu nowej wersji tej struktury układ istniejącej części nie ulega zmianie. Dlatego wyjście musi sprawdzić, czy pole *Version* jest równe lub większe od najniższej wersji, która zawiera pola, które mają być używane przez wyjście.

Jest to pole wejściowe do wyjścia.

Deklaracja 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 */
};
```

Deklaracja języka COBOL (tylko w systemie IBM i)

```
**   MQDXP structure
  10 MQDXP.
**    Structure identifier
   15 MQDXP-STRUCID        PIC X(4).
**    Structure version number
   15 MQDXP-VERSION        PIC S9(9) BINARY.
**    Reserved
   15 MQDXP-EXITOPTIONS    PIC S9(9) BINARY.
**    Application options
   15 MQDXP-APPOPTIONS     PIC S9(9) BINARY.
**    Numeric encoding required by application
   15 MQDXP-ENCODING       PIC S9(9) BINARY.
**    Character set required by application
   15 MQDXP-CODEDCHARSETID PIC S9(9) BINARY.
**    Length in bytes of message data
   15 MQDXP-DATALENGTH     PIC S9(9) BINARY.
**    Completion code
   15 MQDXP-COMPCODE       PIC S9(9) BINARY.
**    Reason code qualifying COMPCODE
   15 MQDXP-REASON         PIC S9(9) BINARY.
**    Response from exit
   15 MQDXP-EXITRESPONSE   PIC S9(9) BINARY.
**    Connection handle
   15 MQDXP-HCONN          PIC S9(9) BINARY.
```

Deklaracja asemblera 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-przekształcanie znaków

Wywołanie MQXCNVC przekształca znaki z jednego zestawu znaków w inny przy użyciu języka programowania C.

To wywołanie jest częścią interfejsu DCI (IBM MQ Data Conversion Interface), który jest jednym z interfejsów środowiska IBM MQ .

Uwaga: Wywołanie może być używane zarówno w środowisku aplikacji, jak i w środowisku wyjścia konwersji danych.

Składnia

MQXCNVC (*Hconn*, *Options*, *SourceCCSID*, *SourceLength*, *SourceBuffer*, *TargetCCSID*, *TargetLength*, *TargetBuffer*, *DataLength*, *CompCode*, *Reason*)

Parametry

hconn
Typ: MQHCONN-input

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

W przypadku wyjścia konwersji danych Hconn jest zwykle uchwytem przekazywanym do wyjścia konwersji danych w polu Hconn struktury MQDXP. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

IBM i W systemie IBM idla parametru Hconnmożna podać następującą wartość specjalną:

ZMQHC_DEF_HCONN
Domyślny uchwyt połączenia.

Jeśli uruchamiana jest aplikacja CICS TS 3.2 lub nowsza, należy upewnić się, że program obsługi wyjścia konwersji znaków, który wywołuje wywołanie MQXCNVC, jest zdefiniowany jako OPENAPI. Ta definicja zapobiega występowaniu błędu MQRC_HCONN_ERROR w 2018 r. spowodowanego niepoprawnym połączeniem i umożliwia zakończenie operacji MQGET.

Opcje
Typ: MQLONG-input

Opcje sterujące działaniem MQXCNVC.

Można podać co najmniej jedną z opisanych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

Domyślna-opcja konwersji: Poniższa opcja steruje użyciem domyślnej konwersji znaków:

MQDCC_DEFAULT_CONVERSION
Konwersja domyślna.

Ta opcja określa, że domyślna konwersja znaków może być używana, jeśli jeden lub oba zestawy znaków określone w wywołaniu nie są obsługiwane. Umożliwia to menedżerowi kolejek użycie podczas konwersji łańcucha domyślnego zestawu znaków określonego przez instalację, który jest zbliżony do określonego zestawu znaków.

Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji łańcucha jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Można tego uniknąć, używając w łańcuchu tylko tych znaków, które są wspólne zarówno dla określonego zestawu znaków, jak i dla domyślnego zestawu znaków.

Domyślne zestawy znaków są definiowane przez opcję konfiguracji podczas instalowania lub restartowania menedżera kolejek.

Jeśli nie określono opcji MQDCC_DEFAULT_CONVERSION, menedżer kolejek używa tylko określonych zestawów znaków do przekształcenia łańcucha i wywołanie kończy się niepowodzeniem, jeśli jeden lub oba zestawy znaków nie są obsługiwane.

Ta opcja jest obsługiwana w następujących środowiskach:

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

Opcja dopełniania: Poniższa opcja umożliwia menedżerowi kolejek dopełnienie przekształconego łańcucha odstępami lub usunięcie nieistotnych znaków końcowych w celu dopasowania przekształconego łańcucha do buforu docelowego:

MQDCC_FILL_TARGET_BUFFER
Wypełnij bufor docelowy.

Ta opcja wymaga, aby konwersja była wykonywana w taki sposób, aby bufor docelowy był całkowicie zapełniony:

- Jeśli podczas konwersji łańcuch jest przekształcany, dodawane są końcowe odstępy w celu wypełnienia buforu docelowego.
- Jeśli łańcuch jest rozwijany podczas konwersji, znaki końcowe, które nie są istotne, są odrzucane, aby przekształcony łańcuch pasował do buforu docelowego. Jeśli można to zrobić pomyślnie, wywołanie kończy się z kodem MQCC_OK i kodem przyczyny MQRC_NONE.

  Jeśli jest zbyt mało nieistotnych znaków końcowych, w buforze docelowym umieszczana jest większość łańcucha, która może się zmieścić, a wywołanie kończy się ostrzeżeniem MQCC_WARNING i kodem przyczyny MQRC_CONVERTED_MSG_TOO_BIG.

  Nieistotne znaki to:

  – Odstępy końcowe
  – Znaki występujące po pierwszym znaku o kodzie zero w łańcuchu (z wyłączeniem samego pierwszego znaku o kodzie zero)
- Jeśli łańcuchy `TargetCCSID`i `TargetLength` są takie, że nie można w pełni ustawić buforu docelowego przy użyciu poprawnych znaków, wywołanie kończy się niepowodzeniem z błędem MQCC_FAILED i kodem przyczyny MQRC_TARGET_LENGTH_ERROR. Taka sytuacja może wystąpić, gdy `TargetCCSID` jest czystym zestawem znaków DBCS (na przykład UTF-16), ale `TargetLength` określa nieparzystą liczbę bajtów.
- Wartość `TargetLength` może być mniejsza lub większa niż `SourceLength`. W przypadku powrotu z MQXCNVC `DataLength` ma taką samą wartość jak `TargetLength`.

Jeśli ta opcja nie jest określona:

- Łańcuch może zwiać lub rozszerzać się w obrębie buforu docelowego zgodnie z wymaganiami. Nieistotne znaki końcowe nie są dodawane ani odrzucane.

  Jeśli przekształcony łańcuch pasuje do buforu docelowego, wywołanie zostanie zakończone z kodem MQCC_OK i kodem przyczyny MQRC_NONE.

Jeśli przekształcony łańcuch jest zbyt duży dla buforu docelowego, w buforze docelowym zostanie umieszczona taka sama część łańcucha, a wywołanie zostanie zakończone ostrzeżeniem MQCC_WARNING i kodem przyczyny MQRC_CONVERTED_MSG_TOO_BIG. W tym przypadku można zwrócić mniej niż `TargetLength` bajtów.

- Wartość `TargetLength` może być mniejsza lub większa niż `SourceLength`. W przypadku powrotu z MQXCNVC wartość `DataLength` jest mniejsza lub równa `TargetLength`.

Ta opcja jest obsługiwana w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

Opcje kodowania: Opisane opcje mogą być używane do określania kodowania liczb całkowitych łańcuchów źródłowych i docelowych. Odpowiednie kodowanie jest używane tylko wtedy, gdy odpowiedni identyfikator zestawu znaków wskazuje, że reprezentacja zestawu znaków w pamięci głównej jest zależna od kodowania używanego dla binarnych liczb całkowitych. Dotyczy to tylko niektórych zestawów znaków wielobajtowych (na przykład zestawów znaków UTF-16).

Kodowanie jest ignorowane, jeśli zestaw znaków jest jednobajtowym zestawem znaków (SBCS) lub wielobajtowym zestawem znaków z reprezentacją w pamięci głównej, która nie jest zależna od kodowania liczb całkowitych.

Należy podać tylko jedną z wartości MQDCC_SOURCE_ * w połączeniu z jedną z wartości MQDCC_TARGET_ *:

MQDCC_SOURCE_ENC_NATIVE
Kodowanie źródłowe jest domyślnym kodowaniem dla środowiska i języka programowania.

MQDCC_SOURCE_ENC_NORMAL
Kodowanie źródłowe jest normalne.

MQDCC_SOURCE_ENC_REVERSED
Kodowanie źródłowe zostało odwrócone.

MQDCC_SOURCE_ENC_NIEZDEFINIOWANY
Kodowanie źródłowe jest niezdefiniowane.

MQDCC_TARGET__ENC_NATIVE
Kodowanie docelowe jest domyślne dla środowiska i języka programowania.

MQDCC_TARGET_ENC_NORMAL
Kodowanie docelowe jest normalne.

MQDCC_TARGET_ENC_REVERSED
Kodowanie docelowe jest odwrócone.

MQDCC_TARGET_ENC_UNDEFINED
Kodowanie docelowe jest niezdefiniowane.

Zdefiniowane wcześniej wartości kodowania można dodać bezpośrednio do pola `Options` . Jeśli jednak kodowanie źródłowe lub docelowe jest uzyskiwane z pola `Encoding` w strukturze MQMD lub innej strukturze, należy wykonać następujące przetwarzanie:

1. Kodowanie na podstawie liczb całkowitych musi zostać wyodrębnione z pola `Encoding` przez wyeliminowanie kodowania liczb zmiennopozycyjnych i upakowanych liczb dziesiętnych. Szczegółowe informacje na ten temat zawiera sekcja “Analizowanie kodowania” na stronie 932 .
2. Kodowanie liczb całkowitych wynikające z kroku 1 musi zostać pomnożone przez odpowiedni współczynnik przed dodaniem do pola `Options` . Czynniki te są następujące:
   - MQDCC_SOURCE_ENC_FACTOR dla kodowania źródłowego
   - MQDCC_TARGET_ENC_FACTOR dla kodowania docelowego

Poniższy kod przykładowy ilustruje, w jaki sposób może być zakodowany w języku programowania C:

```
Options = (MsgDesc.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_SOURCE_ENC_FACTOR
        + (DataConvExitParms.Encoding & MQENC_INTEGER_MASK)
           * MQDCC_TARGET_ENC_FACTOR;
```

Jeśli ta opcja nie zostanie określona, domyślne opcje kodowania będą niezdefiniowane (MQDCC_ * _ENC_UNDEFINED). W większości przypadków nie ma to wpływu na pomyślne zakończenie wywołania MQXCNVC. Jeśli jednak odpowiedni zestaw znaków jest zestawem znaków wielobajtowych z reprezentacją zależną od kodowania (na przykład zestaw znaków UTF-16), wywołanie nie powiedzie się z kodem przyczyny MQRC_SOURCE_INTEGER_ENC_ERROR lub MQRC_TARGET_INTEGER_ENC_ERROR.

Opcje kodowania są obsługiwane w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows
- z/OS

Opcja domyślna: Jeśli nie określono żadnej z opisanych wcześniej opcji, można użyć następującej opcji:

MQDCC_BRAK
Nie określono żadnych opcji.

Parametr MQDCC_NONE jest zdefiniowany w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

SourceCCSID
Typ: MQLONG-input

Jest to identyfikator kodowanego zestawu znaków łańcucha wejściowego w pliku `SourceBuffer`.

SourceLength
Typ: MQLONG-input

Jest to długość łańcucha wejściowego (w bajtach) w `SourceBuffer` ; musi mieć wartość zero lub większą.

SourceBuffer
Typ: MQCHAR x SourceLength -dane wejściowe

Jest to bufor zawierający łańcuch, który ma zostać przekształcony z jednego zestawu znaków na inny.

TargetCCSID
Typ: MQLONG-input

Jest to identyfikator kodowanego zestawu znaków, na który ma zostać przekształcony kod `SourceBuffer` .

TargetLength
Typ: MQLONG-input

Jest to długość buforu wyjściowego w bajtach `TargetBuffer` ; musi mieć wartość zero lub większą. Może być mniejsza lub większa niż `SourceLength`.

TargetBuffer
Typ: MQCHAR x TargetLength -dane wyjściowe

Jest to łańcuch po przekształceniu go w zestaw znaków zdefiniowany przez `TargetCCSID`. Przekształcony łańcuch może być krótszy lub dłuższy niż nieprzekształcony łańcuch. Parametr **`DataLength`** wskazuje liczbę zwróconych poprawnych bajtów.

DataLength
Typ: MQLONG-wyjście

Jest to długość łańcucha zwracanego w buforze wyjściowym `TargetBuffer`. Przekształcony łańcuch może być krótszy lub dłuższy niż nieprzekształcony łańcuch.

CompCode
Typ: MQLONG-wyjście

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny określający `CompCode`.

Jeśli `CompCode` ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli `CompCode` ma wartość MQCC_WARNING:

MQRC_CONVERTED_MSG_TOO_BIG
(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

Jeśli `CompCode` ma wartość MQCC_FAILED:

BŁĄD MQRC_DATA_LENGTH_ERROR
(2010, X'7DA') Niepoprawny parametr długości danych.

BŁĄD MQRC_DBCS_ERROR
(2150, X'866 ') Niepoprawny łańcuch DBCS.

BŁĄD TABELI MQRC_HCONN_ERROR
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Niewystarczające zasoby systemowe.

BŁĄD MQRC_SOURCE_BUFFER_ERROR
(2145, X'861 ') Niepoprawny parametr buforu źródłowego.

BŁĄD MQRC_SOURCE_CCSID_ERROR
(2111, X'83F') Niepoprawny identyfikator źródłowego kodowanego zestawu znaków.

BŁĄD MQRC_SOURCE_INTEGER_ENC_ERROR
(2112, X'840 ') Nierozpoznane kodowanie źródłowe liczby całkowitej.

BŁĄD MQRC_SOURCE_LENGTH_ERROR
(2143, X'85F') Niepoprawny parametr długości źródła.

MQRC_STORAGE_NIEDOSTĘPNY
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

BŁĄD MQRC_TARGET_BUFFER_ERROR
(2146, X'862 ') Niepoprawny parametr buforu docelowego.

BŁĄD MQRC_TARGET_CCSID_ERROR
(2115, X'843 ') Niepoprawny identyfikator kodowanego zestawu znaków.

BŁĄD MQRC_TARGET_INTEGER_ENC_ERROR
(2116, X'844 ') Nie rozpoznano kodowania docelowej liczby całkowitej.

BŁĄD MQRC_TARGET_LENGTH_ERROR
(2144, X'860 ') Niepoprawny parametr długości celu.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów zawiera sekcja Komunikaty i kody przyczyn.

Wywołanie C

```
MQXCNVC (Hconn, Options, SourceCCSID, SourceLength, SourceBuffer,
        TargetCCSID, TargetLength, TargetBuffer, &DataLength,
        &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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 */
```

Deklaracja języka COBOL (tylko w systemie IBM i)

IBM i

```
     CALL 'MQXCNVC' USING HCONN, OPTIONS, SOURCECCSID, SOURCELENGTH,
                          SOURCEBUFFER, TARGETCCSID, TARGETLENGTH,
                          TARGETBUFFER, DATALENGTH, COMPCODE, REASON.
```

Zadeklaruj parametry w następujący sposób:

```
**   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.
```

Deklaracja asemblera S/390

```
         CALL MQXCNVC,(HCONN,OPTIONS,SOURCECCSID,SOURCELENGTH,         X
               SOURCEBUFFER,TARGETCCSID,TARGETLENGTH,TARGETBUFFER,     X
               DATALENGTH,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
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-wyjście konwersji danych

Wywołanie komendy MQ_DATA_CONV_EXIT opisuje parametry, które są przekazywane do wyjścia konwersji danych.

Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQ_DATA_CONV_EXIT (patrz uwaga dotycząca użycia 11).

Ta definicja jest częścią interfejsu DCI (IBM MQ Data Conversion Interface), który jest jednym z interfejsów środowiska IBM MQ .

Składnia

MQ_DATA_CONV_EXIT (*DataConvExitParms*, *MsgDesc*, *InBufferLength*, *InBuffer*, *OutBufferLength*, *OutBuffer*)

Parametry

DataConvExitParms
Typ: MQDXP-input/output

Ta struktura zawiera informacje dotyczące wywołania wyjścia. Wyjście ustawia informacje w tej strukturze w celu wskazania wyniku konwersji. Szczegółowe informacje na temat pól w tej strukturze zawiera sekcja “MQDXP-parametr wyjścia konwersji danych” na stronie 944 .

MsgDesc
Typ: MQMD-wejście/wyjście

Na wejściu do wyjścia jest to deskryptor komunikatu powiązany z danymi komunikatu przekazanymi do wyjścia w parametrze **InBuffer** .

Uwaga: Parametr **MsgDesc** przekazywany do wyjścia jest zawsze najnowszą wersją deskryptora MQMD obsługiwaną przez menedżer kolejek, który wywołuje wyjście. Jeśli wyjście ma być przenośne między różnymi środowiskami, wyjście sprawdzi pole Version w pliku MsgDesc , aby sprawdzić, czy pola, do których wyjście musi uzyskać dostęp, są obecne w strukturze.

W następujących środowiskach do wyjścia przekazywany jest deskryptor MQMD version-2 :

- 

AIX
- 

IBM i
- 

Linux
- 

Windows

We wszystkich innych środowiskach, które obsługują wyjście konwersji danych, do wyjścia jest przekazywana wartość MQMD version-1 .

Jeśli konwersja powiedzie się, wyjście zmieni wartości w polach `Encoding` i `CodedCharSetId` na wartości żądane przez aplikację. Zmiany te zostaną odzwierciedlone w aplikacji. Wszelkie inne zmiany wprowadzone przez wyjście do struktury są ignorowane i nie są odzwierciedlane w aplikacji.

Jeśli wyjście zwraca wartość MQXDR_OK w polu `ExitResponse` struktury MQDXP, ale nie zmienia pól `Encoding` lub `CodedCharSetId` w deskryptorze komunikatu, menedżer kolejek zwraca dla tych pól wartości, które odpowiadały polom w strukturze MQDXP na wejściu do wyjścia.

Długość InBuffer
Typ: MQLONG-input

Długość w bajtach `InBuffer`.

Jest to długość buforu wejściowego `InBuffer`i określa liczbę bajtów, które mają być przetworzone przez wyjście. `InBufferLength` jest mniejszą z długości danych komunikatu przed konwersją oraz długości buforu udostępnianego przez aplikację w wywołaniu MQGET.

Wartość jest zawsze większa od zera.

InBuffer
Typ: MQBYTExInBufferLength -wejście

Bufor zawierający nieprzekształcony komunikat.

Zawiera dane komunikatu przed konwersją. Jeśli wyjście nie może przekształcić danych, menedżer kolejek zwraca treść tego buforu do aplikacji po zakończeniu wyjścia.

Uwaga: Wyjście nie powinno zmieniać `InBuffer` ; Jeśli ten parametr zostanie zmieniony, wyniki są niezdefiniowane.

W języku programowania C parametr ten jest definiowany jako wskaźnik do wartości typu void.

Długość OutBuffer
Typ: MQLONG-input

Długość w bajtach `OutBuffer`.

Jest to długość buforu wyjściowego `OutBuffer`i jest taka sama jak długość buforu udostępnianego przez aplikację w wywołaniu MQGET.

Wartość jest zawsze większa od zera.

OutBuffer
Typ: MQBYTExOutBufferLength -dane wyjściowe

Bufor zawierający przekształcony komunikat.

W przypadku danych wyjściowych z wyjścia, jeśli konwersja zakończyła się pomyślnie (wskazywana przez wartość MQXDR_OK w polu `ExitResponse` parametru **`DataConvExitParms`**), parametr `OutBuffer` zawiera dane komunikatu, które mają zostać dostarczone do aplikacji, w żądanej reprezentacji. Jeśli konwersja nie powiodła się, wszystkie zmiany wprowadzone przez wyjście w tym buforze są ignorowane.

W języku programowania C parametr ten jest definiowany jako wskaźnik do wartości typu void.

Użycie notatek

1. Wyjście konwersji danych jest zapisanym przez użytkownika wyjściem, które odbiera sterowanie podczas przetwarzania wywołania MQGET. Funkcja wykonywana przez wyjście konwersji danych jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami opisanymi w tym miejscu i w powiązanej strukturze parametrów MQDXP.

   Języki programowania, które mogą być używane dla wyjścia konwersji danych, są określane przez środowisko.

2. Wyjście jest wywoływane tylko wtedy, gdy wszystkie poniższe instrukcje są prawdziwe:

   - Opcja MQGMO_CONVERT jest określona w wywołaniu MQGET
   - Pole `Format` w deskryptorze komunikatu nie ma wartości MQFMT_NONE.
   - Komunikat nie znajduje się jeszcze w wymaganej reprezentacji, czyli jeden lub oba elementy `CodedCharSetId` i `Encoding` komunikatu różnią się od wartości określonej przez aplikację w deskryptorze komunikatu dostarczonym w wywołaniu MQGET.
   - Konwersja menedżera kolejek nie została jeszcze zakończona pomyślnie
   - Długość buforu aplikacji jest większa od zera
   - Długość danych komunikatu jest większa od zera
   - Kod przyczyny do tej pory podczas operacji MQGET to MQRC_NONE lub MQRC_TRUNCATED_MSG_ACCEPTED.

3. Podczas zapisywania wyjścia należy rozważyć zakodowanie wyjścia w taki sposób, aby umożliwić mu konwersję obciętych komunikatów. Obcięte komunikaty mogą wystąpić w następujący sposób:

   - Aplikacja odbierająca udostępnia bufor mniejszy niż komunikat, ale określa opcję MQGMO_ACCEPT_TRUNCATED_MSG w wywołaniu MQGET.

     W tym przypadku pole `Reason` w parametrze **DataConvExitParms** na wejściu do wyjścia ma wartość MQRC_TRUNCATED_MSG_ACCEPTED.

   - Nadawca komunikatu obciął go przed wysłaniem. Taka sytuacja może wystąpić w przypadku komunikatów raportu, na przykład (więcej szczegółów zawiera sekcja "Konwersja komunikatów raportu" na stronie 943).

     W tym przypadku pole `Reason` w parametrze **DataConvExitParms** na wejściu do wyjścia ma wartość MQRC_NONE (jeśli aplikacja odbierająca udostępniła bufor wystarczająco duży dla komunikatu).

   Dlatego wartość pola `Reason` na wejściu do wyjścia nie może być zawsze używana do określenia, czy komunikat został obcięty.

   Cechą wyróżniającą obciętego komunikatu jest to, że długość podana dla wyjścia w parametrze **InBufferLength** jest mniejsza niż długość określona przez nazwę formatu zawartą w polu `Format` w deskryptorze komunikatu. Wyjście powinno zatem sprawdzić wartość `InBufferLength` przed podjęciem próby przekształcenia jakichkolwiek danych. Wyjście nie powinno zakładać, że podano pełną ilość danych implikowanych przez nazwę formatu.

   Jeśli wyjście nie zostało zapisane w celu przekształcenia obciętych komunikatów, a wartość `InBufferLength` jest mniejsza niż oczekiwana, wyjście zwróci wartość MQXDR_CONVERSION_FAILED w polu `ExitResponse` parametru **DataConvExitParms** z polami `CompCode` i `Reason` ustawionymi na MQCC_WARNING i MQRC_FORMAT_ERROR.

   Jeśli wyjście zostało zapisane w celu konwersji obciętych komunikatów, wyjście przekształci jak najwięcej danych (patrz uwaga dotycząca następnego użycia), uważając, aby nie próbować badać lub konwertować danych poza koniec pliku `InBuffer`. Jeśli konwersja zakończy się pomyślnie, wyjście pozostawi pole `Reason` w parametrze **DataConvExitParms** bez zmian. Zwraca wartość MQRC_TRUNCATED_MSG_ACCEPTED, jeśli komunikat został obcięty przez menedżer kolejek odbiornika, lub wartość MQRC_NONE, jeśli komunikat został obcięty przez nadawcę komunikatu.

   Możliwe jest również rozwinięcie komunikatu podczas konwersji do punktu, w którym jest on większy niż `OutBuffer`. W takim przypadku wyjście musi zdecydować, czy komunikat ma zostać obcięty.

Pole AppOptions w parametrze **DataConvExitParms** wskazuje, czy aplikacja odbierająca określiła opcję MQGMO_ACCEPT_TRUNCATED_MSG.

4. Na ogół wszystkie dane w komunikacie udostępnionym do wyjścia w produkcie InBuffer są przekształcane lub żadna z nich nie jest przekształcana. Wyjątek od tej reguły występuje jednak wtedy, gdy komunikat jest obcięty przed konwersją lub podczas konwersji; w takim przypadku na końcu buforu może znajdować się niekompletny element (na przykład: 1 bajt znaku dwubajtowego lub 3 bajty 4-bajtowej liczby całkowitej). W takiej sytuacji należy rozważyć pominięcie niekompletnego elementu i ustawienie nieużywanych bajtów w pliku OutBuffer na wartości null. Należy jednak dokonać konwersji kompletnych elementów lub znaków w tablicy lub łańcuchu.
5. Jeśli wyjście jest potrzebne po raz pierwszy, menedżer kolejek próbuje załadować obiekt o takiej samej nazwie jak format (oprócz rozszerzeń). Załadowany obiekt musi zawierać wyjście przetwarzające komunikaty o tej nazwie formatu. Należy rozważyć ustawienie identycznej nazwy wyjścia i nazwy obiektu, który zawiera wyjście, chociaż nie wszystkie środowiska tego wymagają.
6. Nowa kopia wyjścia jest ładowana, gdy aplikacja próbuje pobrać pierwszy komunikat, który używa tego komunikatu Format od czasu połączenia aplikacji z menedżerem kolejek. W przypadku aplikacji CICS lub IMS oznacza to, że podsystem CICS lub IMS jest połączony z menedżerem kolejek. Nową kopię można również załadować w innym czasie, jeśli menedżer kolejek odrzucił wcześniej załadowaną kopię. Z tego powodu wyjście nie może próbować używać pamięci statycznej do przekazywania informacji z jednego wywołania wyjścia do następnego-wyjście może zostać rozładowane między dwoma wywołaniami.
7. Jeśli istnieje wyjście dostarczone przez użytkownika o takiej samej nazwie jak jeden z wbudowanych formatów obsługiwanych przez menedżer kolejek, wyjście dostarczone przez użytkownika nie zastępuje wbudowanej procedury konwersji. Jedynymi okolicznościami, w których takie wyjście jest wywoływane, są:
   - Jeśli wbudowana procedura konwersji nie może obsłużyć konwersji do lub z systemu CodedCharSetId lub Encoding , lub
   - Jeśli nie powiodła się konwersja danych przez wbudowaną procedurę konwersji (na przykład z powodu pola lub znaku, którego nie można przekształcić).
8. Zasięg wyjścia jest zależny od środowiska. Aby zminimalizować ryzyko konfliktów z innymi formatami, należy wybrać nazwy Format . Należy rozważyć rozpoczęcie od znaków, które identyfikują aplikację definiującą nazwę formatu.
9. Wyjście konwersji danych działa w środowisku podobnym do środowiska programu, który wywołał wywołanie MQGET; środowisko zawiera przestrzeń adresową i profil użytkownika (jeśli ma zastosowanie). Program może być agentem kanału komunikatów wysyłającym komunikaty do docelowego menedżera kolejek, który nie obsługuje konwersji komunikatów. Wyjście nie może naruszyć integralności menedżera kolejek, ponieważ nie działa w środowisku menedżera kolejek.
10. Jedynym wywołaniem MQI, które może być używane przez wyjście, jest MQXCNVC. Próba użycia innych wywołań MQI kończy się niepowodzeniem z kodem przyczyny MQRC_CALL_IN_PROGRESS lub innymi nieprzewidywalnymi błędami.
11. Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQ_DATA_CONV_EXIT. Jednak dla nazwy MQ_DATA_CONV_EXIT w języku programowania C podano typedef , którego można użyć do zadeklarowania wyjścia napisanego przez użytkownika w celu upewnienia się, że parametry są poprawne. Nazwa wyjścia musi być taka sama jak nazwa formatu (nazwa zawarta w polu Format w strukturze MQMD), chociaż nie jest to wymagane we wszystkich środowiskach.

   W poniższym przykładzie przedstawiono, w jaki sposób można zadeklarować wyjście przetwarzające format MYFORMAT w języku programowania C:

```
#include "cmqc.h"
#include "cmqxc.h"

MQ_DATA_CONV_EXIT MYFORMAT;

void MQENTRY MYFORMAT(
     PMQDXP   pDataConvExitParms, /* Data-conversion exit parameter
                                     block */
     PMQMD    pMsgDesc,           /* Message descriptor */
```

```
     MQLONG   InBufferLength,     /* Length in bytes of InBuffer */
     PMQVOID  pInBuffer,          /* Buffer containing the unconverted
                                     message */
     MQLONG   OutBufferLength,    /* Length in bytes of OutBuffer */
     PMQVOID  pOutBuffer)         /* Buffer containing the converted
                                     message */
{
  /* C language statements to convert message */
}
```

12. z/OS W systemie z/OS, jeśli wymuszone jest również wyjście przekraczania granicy interfejsu API, jest ono wywoływane po wyjściu konwersji danych.

Wywołanie C

```
exitname (&DataConvExitParms, &MsgDesc, InBufferLength,
         InBuffer, OutBufferLength, OutBuffer);
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
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 */
```

Deklaracja języka COBOL (tylko w systemieIBM i)

IBM i

```
     CALL 'exitname' USING DATACONVEXITPARMS, MSGDESC, INBUFFERLENGTH,
                           INBUFFER, OUTBUFFERLENGTH, OUTBUFFER.
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
**   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).
```

Deklaracja asemblera System/390

```
         CALL EXITNAME,(DATACONVEXITPARMS,MSGDESC,INBUFFERLENGTH,      X
               INBUFFER,OUTBUFFERLENGTH,OUTBUFFER)
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
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
```

Właściwości określone jako elementy MQRFH2

Właściwości deskryptora inne niż komunikat można określić jako elementy w folderach nagłówków MQRFH2 . Przegląd elementów MQRFH2 określanych jako właściwości.

Zachowuje to kompatybilność z wcześniejszymi wersjami klientów IBM MQ JMS i XMS . W tej sekcji opisano sposób określania właściwości w nagłówkach MQRFH2 .

Aby użyć elementów MQRFH2 jako właściwości, należy określić elementy zgodnie z opisem w sekcji Korzystanie z produktu IBM MQ classes for Java . Informacje te uzupełniają informacje opisane w sekcji "MQRFH2 -reguły i nagłówek formatowania 2" na stronie 544.

Odwzorowywanie typów danych właściwości na typy danych MQRFH2

W tym temacie przedstawiono informacje o typach właściwości komunikatu odwzorowanych na odpowiadające im typy danych MQRFH2 .

Tabela 635. Obsługiwane typy danych MQRFH2

Typ właściwości komunikatu	**Typ danych MQRFH2**
MQBYTE []	bin.hex
MQBOOL	boolean (boolowskie)
MQINT8	i1
MQINT16	i2
MQINT32	i4
MQINT64	i8
MQFLOAT32	r4
MQFLOAT64	r8
MQCHAR []	string (łańcuch)

Przyjmuje się, że każdy element bez typu danych ma typ "string".

Typ danych MQRFH2 `int`, czyli liczba całkowita o nieokreślonej wielkości, jest traktowany jak `i8`.

Wartość NULL jest wskazywana przez atrybut elementu `xsi:nil='true'`. Nie należy używać atrybutu `xsi:nil='false'` dla wartości innych niż NULL.

Na przykład następująca właściwość ma wartość null:

```
<NullProperty xsi:nil='true'></NullProperty>
```

Właściwość bajtu lub łańcucha znaków może mieć pustą wartość. Jest on reprezentowany przez element MQRFH2 z wartością elementu o zerowej długości.

Na przykład następująca właściwość ma pustą wartość:

```
<EmptyProperty></EmptyProperty>
```

Obsługiwane foldery MQRFH2

Przegląd użycia pól deskryptora komunikatu jako właściwości.

Foldery <jms>, <mcd>, <mqext>i <usr> są opisane w sekcji Nagłówek MQRFH2 i produkt JMS. Folder <usr> jest używany do transportowania dowolnych właściwości zdefiniowanych przez aplikację JMS , które są powiązane z komunikatem. Grupy nie są dozwolone w folderze <usr> .

Nagłówek MQRFH2 i produkt JMS obsługują następujące dodatkowe foldery:

- <mq>

  Ten folder jest używany i zarezerwowany dla właściwości zdefiniowanych w produkcie MQ, które są używane przez produkt IBM MQ.

- <mq_usr>

  Ten folder może być używany do transportowania dowolnych właściwości zdefiniowanych przez aplikację, które nie są prezentowane jako właściwości zdefiniowane przez użytkownika JMS , ponieważ właściwości te mogą nie spełniać wymagań właściwości JMS . Ten folder może zawierać grupy, których nie może zawierać folder <usr> .

- Dowolny folder oznaczony atrybutem content='properties' .

  Taki folder jest odpowiednikiem folderu <mq_usr> w treści.

- <mqps>

  Ten folder jest używany dla właściwości publikowania/subskrypcji produktu IBM MQ .

Produkt IBM MQ obsługuje również następujące foldery, które są już używane przez serwer WAS/SIB:

- <sib>

  Ten folder jest używany i zarezerwowany dla właściwości komunikatów systemowych WAS/SIB, które nie są ujawnione jako właściwości JMS lub są odwzorowane na właściwości JMS_IBM_ *, ale są ujawnione dla aplikacji WAS/SIB. Są to między innymi właściwości ścieżek routingu skierowanego do przodu i odwrotnego.

  Przynajmniej niektóre z nich nie mogą być prezentowane jako właściwości JMS , ponieważ są tablicami bajtów. Jeśli aplikacja doda właściwości do tego folderu, wartość zostanie zignorowana lub usunięta.

- <sib_usr>

  Ten folder jest używany i zarezerwowany dla właściwości komunikatu użytkownika WAS/SIB, które nie mogą zostać ujawnione jako właściwości użytkownika JMS , ponieważ nie są obsługiwane przez aplikacje WAS/SIB.

  Są to właściwości użytkownika, które można pobrać lub ustawić za pomocą interfejsu SIMessage, ale treść tablicy bajtów jest odwzorowywana na wymaganą wartość właściwości.

  Jeśli aplikacja IBM MQ zapisuje w folderze dowolny element bin.hex , prawdopodobnie otrzyma on plik IOException, ponieważ nie ma oczekiwanego formatu odtwarzania. Jeśli zostanie dodany element inny niż bin.hex , zostanie wyświetlony komunikat ClassCastException.

  Nie należy podejmować próby udostępnienia właściwości dla WAS/SIB przy użyciu tego folderu. W tym celu należy użyć folderu <usr> .

- <sib_context>

  Ten folder jest używany dla właściwości komunikatu systemowego WAS/SIB, które nie są ujawniane w aplikacjach użytkownika WAS/SIB ani jako właściwości JMS . Obejmują one właściwości zabezpieczeń i właściwości transakcyjne, które są używane na potrzeby usług Web Service i podobne.

  Aplikacja nie może dodawać właściwości do tego folderu.

- <mqema>

  Ten folder był używany przez serwer WAS/SIB zamiast folderu <mqext> .

W nazwach folderów MQRFH2 rozróżniana jest wielkość liter.

Następujące foldery są zastrzeżone, niezależnie od kombinacji małych i wielkich liter:

- Dowolny folder z przedrostkiem mq lub wmq ; zarezerwowane do użytku przez IBM MQ.

- Dowolny folder z przedrostkiem `sib` ; zarezerwowane do użytku przez WAS/SIB.
- Foldery `<Root>` i `<Body>` ; zarezerwowane, ale nie używane.

Następujące foldery nie są rozpoznawane jako zawierające właściwości komunikatu:

- `<psc>`

  Używany przez IBM Integration Bus do przekazywania komunikatów komend publikowania/subskrypcji do brokera.

- `<pscr>`

  Używany przez produkt IBM Integration Bus do przechowywania informacji z brokera w odpowiedzi na komunikaty komend publikowania/subskrypcji.

- Dowolny folder, który nie jest zdefiniowany w pliku IBMi nie jest oznaczony atrybutem `content='properties'` .

Nie należy podawać parametru `content='properties'` w folderach `<psc>` ani `<pscr>` . W takim przypadku te foldery są traktowane jako właściwości i prawdopodobnie program IBM Integration Bus przestanie działać zgodnie z oczekiwaniami.

Jeśli aplikacja buduje komunikaty z właściwościami, w nagłówkach MQRFH2 rozpoznawane są jako nagłówek MQRFH2 zawierający właściwości, nagłówek musi znajdować się na liście nagłówków, które można umieścić w łańcuchu na początku komunikatu.

Nagłówek MQRFH2 może być poprzedzony dowolną liczbą standardowych nagłówków MQH, MQCIH, MQDLH, MQIIH, MQTM, MQTMC2lub MQXQH. Łańcuch lub MQCFH kończy analizowanie, ponieważ nie można ich połączyć w łańcuch.

Komunikat może zawierać wiele nagłówków MQRFH2 , które zawierają właściwości komunikatu. Foldery o tej samej nazwie mogą współistnieć w różnych nagłówkach, o ile nie zostanie to ograniczone inaczej, na przykład przez WAS/SIB. Foldery są traktowane jako jeden folder logiczny, jeśli wszystkie znajdują się w znaczących nagłówkach.

Foldery ze znaczących nagłówków nie mogą być scalane z tymi folderami w nieistotnych nagłówkach, ale foldery o takiej samej nazwie w znaczących nagłówkach mogą być scalane, usuwając wszelkie właściwości powodujące konflikt. Aplikacje nie mogą być zależne od układu właściwości w komunikacie.

Grupy MQRFH2 są analizowane pod kątem właściwości w folderach zdefiniowanych przez użytkownika, a nie w folderach `<wmq>`, `<jms>`, `<mcd>`, `<usr>`, `<mqext>`, `<sib>`, `<sib_usr>`, `<sib_context>`i `<mqema>` .

Grupy w folderach właściwości zdefiniowanych przez IBM, z wyjątkiem folderów `<wmq>` i `<mq>` , są analizowane pod kątem właściwości.

Folder MQRFH2 nie może zawierać treści mieszanej; folder lub grupa może zawierać grupy, właściwości lub wartości, ale nie obie te wartości.

Segment komunikatu, pierwszy lub kolejny segment, nie może zawierać właściwości zdefiniowanych przez IBM MQ innych niż te właściwości w deskryptorze komunikatu. Dlatego umieszczenie komunikatu zawierającego takie właściwości z zestawem MQMF_SEGMENT lub MQMF_SEGMENTATION_ALLOWED powoduje, że operacja put kończy się niepowodzeniem z wartością MQRC_SEGMENTATION_NOT_ALLOWED.

Grupy komunikatów mogą jednak zawierać właściwości zdefiniowane przez IBM MQ .

Generowanie nagłówków MQRFH2

Jeśli program IBM MQ przekształca właściwości komunikatu w ich reprezentację w formacie `MQRFH2` , musi dodać do komunikatu parametr `MQRFH2` . Dodaje `MQRFH2` jako oddzielny nagłówek lub scala go z istniejącym nagłówkiem.

Generowanie nowych nagłówków `MQRFH2` przez IBM MQ może zakłócić istniejące nagłówki w komunikacie. Aplikacje analizujące bufor komunikatów dla nagłówków muszą mieć świadomość, że liczba i pozycja nagłówków w buforze może w pewnych okolicznościach ulec zmianie. Program IBM

MQ próbuje zminimalizować wpływ dodawania właściwości do komunikatu przez scalenie właściwości komunikatu z istniejącym nagłówkiem MQRFH2 , gdzie jest to możliwe. Próbuje również zminimalizować wpływ, wstawiając wygenerowany kod MQRFH2 do stałej pozycji względem innych nagłówków w buforze komunikatów.

Wygenerowany nagłówek MQRFH2 jest umieszczany po nagłówku MQMDi dowolnej liczbie nagłówków MQXQH, MQRFHi MQDLH , niezależnie od ich kolejności. Wygenerowany nagłówek MQRFH2 jest umieszczany bezpośrednio przed pierwszym nagłówkiem, który nie jest nagłówkiem MQMD, MQXQH, MQDLHani MQRFH .

z/OS W systemach z/OS wygenerowany nagłówek MQRFH2 jest tworzony w pliku CCSIDaplikacji. Jest to zdefiniowane w następujący sposób:

- W przypadku wsadowych aplikacji LE używających interfejsu DLL parametr CCSID jest CODESET powiązany z bieżącymi ustawieniami narodowymi w momencie wydania komendy **MQCONN** (wartość domyślna to 1047).
- W przypadku aplikacji wsadowych LE powiązanych z jednym z wsadowych kodów pośredniczących MQ wartość CCSID jest CODESET powiązana z bieżącymi ustawieniami narodowymi w czasie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku aplikacji wsadowych innych niż LE działających w wątku z/OS UNIX System Services (z/OS UNIX) CCSID jest wartością THLICCSID w czasie pierwszego wywołania MQI wydanego po **MQCONN** (wartość domyślna to 1047).
- W przypadku innych aplikacji wsadowych CCSID to CCSID menedżera kolejek.

W przypadku aplikacji LE ustawienia narodowe można zmienić za pomocą wywoływalnej usługi setlocale() / CEESETL LE . W przypadku aplikacji innych niż LE działających w wątkach z/OS UNIX wartość zmiennej THLICCSID można zmienić za pomocą z/OS UNIX makra odwzorowania **BPXYTHLI**.

Reguły scalania wygenerowanych MQRFH2

Poniższe reguły mają zastosowanie do scalania wygenerowanego MQRFH2 z istniejącym MQRFH2. Wygenerowany nagłówek MQRFH2 jest scalany z istniejącym nagłówkiem MQRFH2 , jeśli:

1. Istniejący plik MQRFH2 znajduje się w tej samej pozycji, w której IBM MQ umieszcza wygenerowany plik MQRFH2lub wcześniejszy w łańcuchu nagłówków.
2. Identyfikator CCSID wygenerowanych właściwości jest taki sam, jak identyfikator NameValueCCSID istniejącego identyfikatora MQRFH2.

W przeciwnym razie wygenerowany nagłówek jest umieszczany oddzielnie w buforze, w pozycji opisanej wcześniej.

Reguły scalania folderów w istniejącym MQRFH2

Jeśli właściwości komunikatu zostaną scalone z istniejącym MQRFH2, istniejący MQRFH2 jest skanowany w poszukiwaniu folderów, które są zgodne z właściwościami komunikatu, i scala je. Jeśli zgodny folder nie istnieje, nowy folder jest dodawany na końcu istniejących folderów. Jeśli zgodny folder istnieje, jest on przeszukiwany. Wszystkie zgodne właściwości są nadpisywane. Wszystkie nowe są dodawane na końcu folderu.

Ograniczenia dotyczące folderów MQRFH2

Przegląd ograniczeń dotyczących folderów w nagłówkach MQRFH2

Ograniczenia MQRFH2 dotyczą następujących folderów:

- Nazwy elementów w folderze <usr> nie mogą zaczynać się od przedrostka JMS ; Takie nazwy właściwości są zarezerwowane do użytku przez JMS i nie są poprawne dla właściwości zdefiniowanych przez użytkownika.

  Taka nazwa elementu nie powoduje niepowodzenia analizowania struktury MQRFH2 , ale nie jest dostępna dla interfejsów API właściwości komunikatu produktu IBM MQ .

- Nazwy elementów w folderze <usr> nie mogą zawierać małych i wielkich liter, wartości NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, IS ani ESCAPE. Te nazwy są zgodne ze słowami kluczowymi SQL i utrudniają analizowanie selektorów, ponieważ folder <usr> jest domyślnym folderem używanym, gdy nie określono folderu dla konkretnej właściwości w selektorze.

  Taka nazwa elementu nie powoduje niepowodzenia analizowania struktury MQRFH2 , ale nie jest dostępna dla interfejsów API właściwości komunikatu produktu IBM MQ .
- Model zawartości folderu <usr> jest następujący:
  - Jako nazwy elementu można użyć dowolnej poprawnej nazwy XML, pod warunkiem, że nie zawiera ona dwukropka.
  - Dozwolone są tylko proste elementy, a nie foldery zagnieżdżone.
  - Wszystkie elementy przyjmują domyślny typ łańcucha, o ile nie zostaną zmodyfikowane przez atrybut dt="xxx" .
  - Wszystkie składniki są opcjonalne, ale nie powinny występować w folderze więcej niż raz.
- Nazwy elementów w dowolnym folderze uważanym za zawierający właściwości komunikatu nie mogą zawierać kropki (.) (Znak Unicode U+002E), ponieważ jest on używany w nazwach właściwości w celu wskazania hierarchii.

  Taka nazwa elementu nie powoduje niepowodzenia analizowania struktury MQRFH2 , ale nie jest dostępna dla interfejsów API właściwości komunikatu produktu IBM MQ .

Ogólnie rzecz biorąc, nagłówki MQRFH2 , które zawierają poprawne dane w stylu XML, mogą być analizowane przez produkt IBM MQ bez niepowodzenia, chociaż niektóre elementy nagłówka MQRFH2 nie są dostępne za pośrednictwem interfejsów API właściwości komunikatu IBM MQ .

Konflikty nazw elementów MQRFH2

Przegląd konfliktów w nazwach elementów MQRFH2 .

Do właściwości komunikatu można przyłączyć tylko jedną wartość. Jeśli próba uzyskania dostępu do właściwości prowadzi do konfliktu wartości, jedna z nich jest wybierana w preferencjach zamiast innej.

Składnia komendy IBM MQ na potrzeby uzyskiwania dostępu do elementów MQRFH2 umożliwia unikalną identyfikację elementu, jeśli folder nie zawiera elementów o takiej samej nazwie. Jeśli folder zawiera więcej niż jeden element o takiej samej nazwie, wartość użytej właściwości jest najbliższa wartości nagłówka komunikatu.

Ma to zastosowanie, jeśli co najmniej dwa foldery o tej samej nazwie są zawarte w różnych istotnych nagłówkach MQRFH2 w obrębie tego samego komunikatu.

Konflikt może wystąpić, gdy wywołanie MQGET zostanie przetworzone po dwukrotnym ustawieniu właściwości deskryptora innego niż komunikat: zarówno przez wywołanie MQSETMP, jak i bezpośrednio w surowym nagłówku MQRFH2 .

W takim przypadku właściwość powiązana z komunikatem przez wywołanie interfejsu API ma pierwszeństwo przed właściwością w danych komunikatu, czyli właściwością w nieprzetworzonym nagłówku MQRFH2 . Jeśli wystąpi konflikt, jest on traktowany jako logicznie występujący przed danymi komunikatu.

Odwzorowanie nazw właściwości na nazwy folderów i elementów MQRFH2

Przegląd różnic między nazwami właściwości i nazwami elementów w nagłówku MQRFH2 .

Jeśli używane są zdefiniowane interfejsy API, które generują nagłówki MQRFH2 , w celu określenia właściwości komunikatu (na przykład MQ JMS), nazwa właściwości nie musi być nazwą elementu w folderze MQRFH2 .

Oznacza to, że odwzorowanie odbywa się z nazwy właściwości na element MQRFH2 i w odwrotnym kierunku, z uwzględnieniem zarówno nazwy folderu, który zawiera element, jak i nazwy elementu. Niektóre przykłady z serwisu IBM MQ classes for JMS zostały już opisane w sekcji Korzystanie z programu IBM MQ classes for Java.

Tabela 636. Nazwy właściwości odwzorowane na nazwy folderów i elementów MQRFH2		
Nazwa właściwości	**Nazwa folderu MQRFH2**	**Nazwa elementu MQRFH2**
JMSDestination	jms	Dst
JMSType	mcd	Type, Set, Fmt
xxx (zdefiniowany przez użytkownika, gdzie xxx nie rozpoczyna się od JMS)	usr	xxx

Oznacza to, że gdy aplikacja JMS uzyskuje dostęp do właściwości JMSDestination , jest ona odwzorowywana na element Dst w folderze <jms> .

Podczas określania właściwości jako elementów MQRFH2 produkt IBM MQ definiuje swoje elementy w następujący sposób:

Tabela 637. Nazwy właściwości odwzorowane na nazwy folderów, grup i elementów MQRFH2			
Nazwa właściwości	**Nazwa folderu MQRFH2**	**Nazwa grupy MQRFH2**	**Nazwa elementu MQRFH2**
<Property>	<usr>	nie dotyczy	<Property>
<folder>. <Property>	<folder>	nie dotyczy	<Property>
<folder>. <group>. <Property>	<folder>	<group>	<Property>

Na przykład, gdy aplikacja IBM MQ próbuje uzyskać dostęp do właściwości Property1 , jest ona odwzorowywana na element Property1 w folderze <usr> . Właściwość wmq . Property2 jest odwzorowywana na właściwość Property2 w folderze <wmq> .

Jeśli nazwa właściwości zawiera więcej niż jedną nazwę. , używana jest nazwa elementu MQRFH2 , która występuje po ostatnim. Grupy MQRFH2 są używane do tworzenia hierarchii; zagnieżdżone grupy MQRFH2 są dozwolone.

Dostęp do właściwości nagłówka JMS i specyficznych dla dostawcy, które są zawarte w MQRFH2 w folderach <mcd>, <jms>i <mqext> , jest uzyskiwany przez aplikację IBM MQ przy użyciu nazw skróconych zdefiniowanych w sekcji Korzystanie z produktu IBM MQ classes for Java .

Dostęp do zdefiniowanych przez użytkownika właściwości JMS można uzyskać z folderu <usr> . Aplikacja IBM MQ może używać folderu <usr> dla swoich właściwości aplikacji, jeśli jest ona akceptowalna dla aplikacji JMS jako jedna z jej właściwości zdefiniowanych przez użytkownika.

Jeśli nie jest to akceptowalne, wybierz inny folder. Folder <wmq_usr> jest udostępniany jako standardowe położenie dla takich właściwości innych niżJMS .

Aplikacje mogą określać i używać dowolnego folderu MQRFH2 z dobrze zdefiniowanym użyciem, co nie zostało udokumentowane w sekcji “Właściwości określone jako elementy MQRFH2” na stronie 961 , jeśli zostały spełnione następujące warunki:

1. Folder może być już używany lub może być używany w przyszłości przez inną aplikację zapewniającą niezdefiniowany dostęp do zawartych w nim właściwości. Sugerowaną konwencję nazewnictwa dla nazw właściwości można znaleźć w sekcji Nazwy właściwości .
2. Właściwości nie są dostępne dla wcześniejszych wersji klienta IBM MQ classes for JMS lub XMS , które mogą uzyskać dostęp tylko do folderu <usr> dla właściwości zdefiniowanych przez użytkownika.
3. Folder musi być oznaczony atrybutem content o wartości properties, na przykład content='properties'.

   Produkt “MQSETMP-ustawianie właściwości komunikatu” na stronie 805 automatycznie dodaje ten atrybut zgodnie z wymaganiami. Tego atrybutu nie można dodawać do żadnego z folderów zdefiniowanych przez IBM, na przykład <jms> i <usr>. Powoduje to, że komunikat jest odrzucany przez klienta IBM MQ classes for JMS przed IBM WebSphere MQ 7.0. za pomocą MessageFormatException.

Ponieważ folder <usr> jest domyślnym położeniem właściwości o składni <Property> , aplikacja IBM MQ i aplikacja JMS uzyskują dostęp do tej samej wartości właściwości zdefiniowanej przez użytkownika przy użyciu tej samej nazwy.

Zastrzeżone nazwy folderów

Istnieje kilka zastrzeżonych nazw folderów. Nie można używać takich nazw, jak przedrostki folderów; na przykład nazwa `Root.Property1` nie ma dostępu do poprawnej właściwości, ponieważ nazwa `Root` jest zastrzeżona. Następująca lista zawiera zastrzeżone nazwy folderów:

- Element główny
- Treść
- Właściwości
- Środowisko
- LocalEnvironment
- DestinationList
- ExceptionList
- InputBody
- InputRoot
- InputProperties
- Środowisko InputLocal
- Lista InputDestination
- Lista InputException
- OutputRoot
- Środowisko OutputLocal
- Lista OutputDestination
- Lista OutputException

Odwzorowywanie pól deskryptora właściwości na nagłówki MQRFH2

Gdy właściwość jest przekształcana w element MQRFH2 , do określenia istotnych pól deskryptora właściwości używane są następujące atrybuty elementu: opisuje sposób translacji pól MQPD na atrybuty elementu MQRFH2 .

Obsługa

Pole deskryptora właściwości obsługi jest podzielone na trzy atrybuty elementu

- Atrybut elementu **sr** określa wartości w masce bitowej MQPD_REJECT_UNSUP_MASK.
- Atrybut elementu **sa** określa wartości w masce bitowej MQPD_ACCEPT_UNSUP_MASK.
- Atrybut elementu **sx** określa wartości w masce bitowej MQPD_ACCEPT_UNSUP_IF_XMIT_MASK.

Te atrybuty elementu są poprawne tylko w folderze < mq> i są ignorowane, jeśli zostaną ustawione dla elementów w innych folderach zawierających właściwości.

Tabela 638. Pola MQPD odwzorowane na atrybuty elementu MQRFH2

Wartość wsparcia	Element MQRFH2 , atrybut	MQRFH2 , wartość atrybutu
MQPD_SUPPORT_OPTIONAL	sa	opcjonalne Jest to wartość domyślna.
MQPD_SUPPORT_REQUIRED,	SR	wymagane
MQPD_SUPPORT_REQUIRED_IF_LOCAL	sx	lokalne

Kontekst

Atrybut elementu **context** służy do wskazywania kontekstu komunikatu, do którego należy właściwość. Użyj tylko jednej wartości. Ten atrybut elementu jest poprawny dla właściwości w dowolnym folderze zawierającym właściwości.

Tabela 639. Wartości kontekstu odwzorowane na wartości atrybutu MQRFH2

Wartość kontekstu	**MQRFH2 , wartość atrybutu**
MQPD_NO_CONTEXT	brak Jest to wartość domyślna.
MQPD_USER _KONTEKST	użytkownik

CopyOptions

Atrybut elementu **copy** służy do wskazywania komunikatów, do których ma zostać skopiowana właściwość. Akceptowalna jest więcej niż jedna wartość; wiele wartości należy rozdzielić przecinkiem. Na przykład zarówno **copy='reply'** , jak i **copy='publish,report'** są poprawne. Ten atrybut elementu jest poprawny dla właściwości w dowolnym folderze zawierającym właściwości.

Uwaga: W definicji atrybutu można używać pojedynczych lub podwójnych cudzysłowów, na przykład **copy='reply'** lub **copy="report"**

Tabela 640. Wartości opcji CopyOption odwzorowane na wartości atrybutów MQRFH2

Wartość CopyOption	**MQRFH2 , wartość atrybutu**
MQPD_KOPIUJ_DALEJ	postępująca
MQPD_COPY_REPLY	reply
MQPD_COPY_REPORT	raportowanie
MQPD_COPY_PUBLISH	publikować
MQPD_COPY_ALL	wszystkie Nie należy podawać tej wartości z żadną inną wartością. W przypadku użycia z inną wartością ma ona pierwszeństwo przed dowolną wartością z wyjątkiem **none**.
MQPD_COPY_DEFAULT	default Jest to wartość domyślna. Jest to równoważne określeniu trzech wartości: MQCOPY_FORWARD, MQCOPY_REPORT i MQCOPY_PUBLISH. Nie należy podawać tej wartości z żadną inną wartością.
MQPD_COPY_NONE	brak Nie należy podawać tej wartości z żadną inną wartością. W przypadku użycia z inną wartością ma to pierwszeństwo.

Ograniczenia dotyczące folderu < mq> MQRFH2

Gdy komunikat jest umieszczany w kolejce, jest przeszukiwany pod kątem folderu < mq>, aby komunikat mógł zostać przetworzony zgodnie z jego właściwościami zdefiniowanymi w produkcie MQ. Aby umożliwić wydajne analizowanie właściwości zdefiniowanych w produkcie MQ, do folderu mają zastosowanie następujące ograniczenia:

- Tylko właściwości w pierwszym znaczącym folderze < mq> w komunikacie są wykonywane przez produkt MQ. Właściwości w dowolnym innym folderze < mq> w komunikacie są ignorowane.

- Jeśli folder jest w formacie UTF-8, w folderze dozwolone są tylko jednobajtowe znaki UTF-8 . Wielobajtowy znak w folderze może spowodować niepowodzenie analizowania i odrzucenie komunikatu.
- Nie należy dołączać grup MQRFH2 do folderu < mq>. Obecność znaku Unicode U+003C w wartości właściwości spowoduje odrzucenie komunikatu.
- W folderze nie należy używać łańcuchów o zmienionym znaczeniu. Łańcuch o zmienionym znaczeniu jest traktowany jako rzeczywista wartość elementu.
- Tylko znak Unicode U+0020 jest traktowany jako biały znak w folderze. Wszystkie pozostałe znaki są traktowane jako istotne i mogą spowodować niepowodzenie analizowania folderu oraz odrzucenie komunikatu.

Jeśli analizowanie folderu < mq> nie powiedzie się lub folder nie będzie przestrzegać tych ograniczeń, komunikat zostanie odrzucony z kodem CompCode **MQCC_FAILED** i przyczyną **MQRC_RFH_RESTRICTED_FORMAT_ERR**.

Niepoprawne nagłówki MQRFH2

W czasie przetwarzania wywołań MQPUT, MQPUT1lub MQGET może nastąpić częściowa analiza dowolnych nagłówków MQRFH2 w komunikacie w celu sprawdzenia, które foldery są uwzględnione, oraz w celu określenia, czy foldery zawierają właściwości. Przegląd nagłówków MQRFH2 , które nie są poprawne.

Jeśli częściowe analizowanie komunikatu nie może zakończyć się pomyślnie, ponieważ struktura nie jest poprawna, na przykład pole `StrucLength` jest zbyt małe, wykonaj następujące czynności:

- Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_RFH_ERROR, jeśli można określić, że aplikacja zawiera opcję IBM WebSphere MQ 7 , dzięki czemu istniejące aplikacje nie kończą się niepowodzeniem.
- Wywołanie MQGET zostało pomyślnie zwrócone, a komenda MQRFH2 zawierająca błąd została zwrócona w podanym buforze.

Jeśli częściowe analizowanie nie powiedzie się, ponieważ nie można wykryć, czy konkretny folder zawiera właściwości, czy nie, na przykład folder rozpoczyna się od `<<jms`, więc analizowanie nie powiedzie się przed ustaleniem nazwy folderu:

- Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny MQRC_RFH_FORMAT_ERROR, jeśli można określić, że aplikacja zawiera opcję IBM WebSphere MQ 7 , dzięki czemu istniejące aplikacje nie kończą się niepowodzeniem.
- Wywołanie MQGET zostało pomyślnie zwrócone, a komenda MQRFH2 zawierająca błąd została zwrócona w podanym buforze.
- Podczas wewnętrznego działania menedżera kolejek komunikat nie jest odrzucany z powodu niepoprawnie sformatowanego folderu, ale folder jest zawsze traktowany tak, jakby w nim nie znajdowały się żadne właściwości.

Komunikat może przepływać przez sieć menedżera kolejek z folderem zawierającym taki błąd składniowy, ale nigdy nie jest analizowany i wykrywany, podczas gdy jeden lub więcej folderów w komunikacie to:

- Ważne
- Pomyślnie przeanalizowano
- Używane podczas przetwarzania komunikatu

Dlatego wykrywanie nie jest gwarantowane.

Jeśli jedna z aplikacji używa “MQSETMP-ustawianie właściwości komunikatu” na stronie 805lub MQINQMP w celu uzyskania dostępu do właściwości, co powoduje, że folder MQRFH2 jest w pełni analizowany, co powoduje wykrycie błędu, który nie może zostać zakończony, jest wskazywane przez odpowiedni kod powrotu do wywołania API. Żadne właściwości w folderze nie są udostępniane aplikacji.

Jeśli zostanie podjęta próba pełnego przeanalizowania folderu MQRFH2 , a analizator składni znajdzie nierozpoznane atrybuty elementu lub nierozpoznany typ danych, analizowanie będzie kontynuowane i zakończone pomyślnie bez generowania ostrzeżeń. Nie stanowi to błędu analizowania.

konwersja stron kodowych

W tej sekcji opisano nazwy zestawów kodowych i identyfikatory CCSID, język narodowy, obsługę konwersji konwersja z/OS , konwersja IBM i , i Unicode.

Każda sekcja w języku narodowym zawiera następujące informacje:

- Obsługiwane rodzime identyfikatory CCSID
- Nieobsługiwane konwersje stron kodowych

W informacjach tych używane są następujące terminy:

AIX
Wskazuje IBM MQ for AIX.

Linux
Zawiera wartość IBM MQ dla Linux dla Intel i IBM MQ dla Linux dla zSeries.

OS/400
Wskazuje IBM MQ for IBM i.

Windows
Wskazuje IBM MQ for Windows.

z/OS
Wskazuje IBM MQ for z/OS.

Domyślnie konwersja danych jest wykonywana w systemie docelowym (odbierającym).

Jeśli produkt źródłowy obsługuje konwersję, kanał można skonfigurować i wymieniać dane, ustawiając atrybut kanału CONVERT na wartość YES w źródle.

Uwaga:

1. Konwersja informacji o systemie IBM MQ MQI client odbywa się na serwerze, dlatego serwer musi obsługiwać konwersję z identyfikatora CCSID klienta na identyfikator CCSID serwera.
2. Konwersja może obejmować obsługę dodaną przez CSD/PTF do najnowszej wersji systemu IBM MQ. Sprawdź zawartość najnowszego poziomu usług, aby sprawdzić, czy konieczne jest zainstalowanie CSD/PTF w celu włączenia tej konwersji.
3. Identyfikator CCSID menedżera kolejek systemu IBM MQ musi mieć wartość Mixed (mieszany) lub SBCS (SBCS).
4. Niektóre identyfikatory CCSID, na przykład 850 w systemie AIX, które nie są obsługiwane przez system operacyjny, mogą być nadal używane przez aplikację oraz mogą być ustawione jako identyfikator CCSID menedżera kolejek systemu IBM MQ . Jest to dozwolone tylko w celu zapewnienia kompatybilności wstecznej i konwersja nie powiedzie się, jeśli odpowiednie tabele konwersji nie zostaną zainstalowane.

Więcej informacji na temat odniesień między niektórymi numerami identyfikatorów CCSID i nazwami zestawów kodowych w branży zawiera sekcja Tabela 641 na stronie 971 .

Odsyłacze pokrewne

“Języki narodowe” na stronie 971
Te informacje zawierają języki obsługiwane przez produkt IBM MQ.

Nazwy zestawów kodowych i identyfikatory CCSID

Nazwy zestawów kodowych i odpowiadające im identyfikatory CCSID dla każdej nazwy zestawu kodowego.

z/OS Produkt IBM MQ for z/OS udostępnia więcej konwersji niż wymieniono w tabelach specyficznych dla języka. Pełną listę konwersji zawiera sekcja Tabela 674 na stronie 997.

Tabela 641. Nazwy zestawów kodowych i identyfikatory CCSID

Nazwy zestawów kodowych	**Identyfikatory CCSID**
ISO 8859-1	819
ISO 8859-2	912
ISO 8859-3	913
ISO 8859-5	915
ISO 8859-6	1089
ISO 8859-7	813
ISO 8859-8	916
ISO 8859-9	920
ISO 8859-13	921
ISO 8859-15 (euro)	923
big5	950
eucJP	954 5050 33722
eucKR	970
eucTW	964
eucCN	1383
PCK,	943
GBk	1386
koi8-r	878

Języki narodowe

Te informacje zawierają języki obsługiwane przez produkt IBM MQ.

IBM MQ obsługuje następujące języki:

- Angielski (Stany Zjednoczone)-patrz temat “Angielski (Stany Zjednoczone)” na stronie 972
- Niemiecki-patrz temat “niemiecki” na stronie 973
- Duński i norweski-patrz temat “Duński i norweski” na stronie 973
- Fiński i szwedzki-patrz temat “Fiński i szwedzki” na stronie 974
- Włoski-patrz temat “włoski” na stronie 975
- Hiszpański-patrz temat “hiszpański” na stronie 976
- Angielski (Wielka Brytania)/Gaelic-patrz temat “angielski (Wielka Brytania) /gaelicki” na stronie 976
- Francuski-patrz temat “francuski” na stronie 977
- Wielojęzyczny-patrz temat “Wielojęzyczne” na stronie 978
- Portugalski-patrz temat “portugalski” na stronie 978
- Islandzki-patrz temat “islandzki” na stronie 979
- Języki wschodnioeuropejskie-patrz temat “Języki wschodnioeuropejskie” na stronie 980
- Cyrylica-patrz temat “cyrylica” na stronie 981

- Estoński-patrz temat “estoński” na stronie 982
- Łotewski i litewski-patrz temat “Łotewski i litewski” na stronie 983
- Ukraiński-patrz temat “ukraiński” na stronie 984
- Grecki-patrz temat “grecki” na stronie 984
- Turecki-patrz temat “turecki” na stronie 985
- Hebrajski-patrz temat “hebrajski” na stronie 986
- Farsi-patrz temat “farsi” na stronie 988
- Urdu-patrz temat “urdu” na stronie 988
- Tajski-patrz temat “tajski” na stronie 988
- Laotań-patrz temat “laotański” na stronie 989
- Wietnamski-patrz temat “wietnamski” na stronie 989
- Japoński Latin SBCS-patrz temat “japoński łaciński SBCS” na stronie 990
- japoński Katakana SBCS-patrz temat “japoński Katakana SBCS” na stronie 991
- Japoński Kanji/Latin Mixed-patrz temat “Japoński Kanji/łaciński mieszany” na stronie 992
- Japoński Kanji/Katakana Mixed-patrz temat “Japoński Kanji/Katakana Mieszany” na stronie 993
- Koreański-patrz temat “koreański” na stronie 995
- Chiński uproszczony-patrz temat “chiński uproszczony” na stronie 995
- Chiński tradycyjny-patrz temat “chiński tradycyjny” na stronie 996

Angielski (Stany Zjednoczone)

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka angielskiego (Stany Zjednoczone).

Tabela 642. Rodzime identyfikatory CCSID dla języka angielskiego (Stany Zjednoczone) na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	37, 924, 1140
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 1252, 5348, 858
Linux Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

37
Nie konwertuje na strony kodowe 923, 858

924
Nie konwertuje na strony kodowe 437, 858, 1051, 1140, 1252, 1275, 5348

1140
Nie konwertuje na strony kodowe 924, 1051, 1275

niemiecki
Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka niemieckiego.

Tabela 643. Rodzime identyfikatory CCSID dla języka niemieckiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	273, 924, 1141
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i
IBM i

Strona kodowa:

273
Nie konwertuje na strony kodowe 858, 923, 924, 1275

924
Nie konwertuje na strony kodowe 273, 437, 858, 1051, 1141, 1252, 1275, 5348

1141
Nie konwertuje na strony kodowe 924, 1051, 1275

Duński i norweski
Szczegóły dotyczące CCSID i konwersji CCSID dla języka duńskiego i norweskiego.

Tabela 644. rodzime identyfikatory CCSID dla języka duńskiego i norweskiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	277, 924, 1142
AIX	819, 923, 5348
Windows	850, 858, 865, 1252, 5348
Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

277
Nie konwertuje na strony kodowe 858, 923, 924, 1275

924
Nie konwertuje na strony kodowe 277, 858, 865, 1051, 1142, 1252, 1275, 5348

1142
Nie konwertuje na strony kodowe 924, 865, 1051, 1275

AIX

AIX

Strona kodowa:

819
Nie konwertuje na stronę kodową 865

Windows

Windows

Strona kodowa:

865
Nie konwertuje na strony kodowe 1051, 1275

Fiński i szwedzki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka fińskiego i szwedzkiego.

Tabela 645. Rodzime identyfikatory CCSID dla języka fińskiego i szwedzkiego na obsługiwanych platformach

Platforma	**Rodzime identyfikatory CCSID**
IBM i IBM i z/OS z/OS	278, 924, 1143
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 865, 1252, 5348
Linux Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

278
Nie konwertuje na strony kodowe 858, 923, 924, 1275

924
Nie konwertuje na strony kodowe 278, 437, 858, 865, 1051, 1143, 1252, 1275, 5348

1143
Nie konwertuje na strony kodowe 865, 924, 1051, 1275

AIX

Strona kodowa:

819
Nie konwertuje na stronę kodową 865

850
Nie konwertuje na stronę kodową 865

Windows

Strona kodowa:

865
Nie konwertuje na strony kodowe 1051, 1275

włoski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka włoskiego.

Tabela 646. Rodzime identyfikatory CCSID dla języka włoskiego na obsługiwanych platformach

Platforma	**Rodzime identyfikatory CCSID**
 IBM i  z/OS	280, 924, 1144
 AIX	819, 923, 5348
 Windows	437, 850, 858, 1252, 5348
 Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

280
Nie konwertuje na strony kodowe 858, 923, 924, 1275

924
Nie konwertuje na strony kodowe 280, 437, 858, 1051, 1144, 1252, 1275, 5348

1144
Nie konwertuje na strony kodowe 924, 1051, 1275

hiszpański

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka hiszpańskiego.

Tabela 647. Rodzime identyfikatory CCSID dla języka hiszpańskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i z/OS	284, 924, 1145
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

284
Nie konwertuje na strony kodowe 858, 923, 924, 1275

924
Nie konwertuje na strony kodowe 284, 437, 858, 1051, 1145, 1252, 1275, 5348

1145
Nie konwertuje na strony kodowe 924, 1051, 1275

angielski (Wielka Brytania) /gaelicki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka angielskiego/gaelickiego w Wielkiej Brytanii.

Tabela 648. Rodzime identyfikatory CCSID dla języka angielskiego/gaelickiego w Wielkiej Brytanii na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i z/OS	285, 924, 1146
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348

Tabela 648. Rodzime identyfikatory CCSID dla języka angielskiego/gaelickiego w Wielkiej Brytanii na obsługiwanych platformach (kontynuacja)	
Platforma	**Rodzime identyfikatory CCSID**
Linux Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

285
Nie konwertuje na strony kodowe 858, 923, 924, 1275

924
Nie konwertuje na strony kodowe 285, 437, 858, 1051, 1146, 1252, 1275, 5348

1146
Nie konwertuje na strony kodowe 924, 1051, 1275

francuski

Szczegóły dotyczące CCSID i konwersji CCSID dla języka francuskiego.

Tabela 649. Rodzime identyfikatory CCSID dla języka francuskiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i IBM i z/OS z/OS	297, 924, 1147
AIX AIX	819, 923, 5348
Windows Windows	437, 850, 858, 1252, 5348
Linux Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

297
Nie konwertuje na strony kodowe 858, 923, 924, 1275, 5348

924
Nie konwertuje na strony kodowe 297, 437, 858, 1051, 1147, 1252, 1275, 5348

1147
Nie konwertuje na strony kodowe 924, 1051, 1275

Wielojęzyczne

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języków wielojęzycznych.

Tabela 650. Rodzime identyfikatory CCSID dla konwersji wielojęzycznej na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	500, 924, 1148
AIX	819, 923, 5348
Windows	437, 850, 858, 1252, 5348
Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

500
Nie konwertuje na strony kodowe 858, 923

924
Nie konwertuje na strony kodowe 437, 858, 1051, 1148, 1252, 1275, 5348

1148
Nie konwertuje na strony kodowe 924, 1051, 1275

portugalski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka portugalskiego.

Tabela 651. Rodzime identyfikatory CCSID dla języka portugalskiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i	37, 500, 924, 1140
z/OS	500, 924, 1140
AIX	819, 923, 5348
Windows	850, 858, 860, 1252, 5348
Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

37
Nie konwertuje na strony kodowe 858, 923, 1275

500
Nie konwertuje na strony kodowe 858, 923, 1275

924
Nie konwertuje na strony kodowe 858, 860, 1051, 1140, 1252, 1275, 5348

1140
Nie konwertuje na strony kodowe 860, 924, 1051, 1275

Windows

Strona kodowa:

860
Nie konwertuje na strony kodowe 1051, 1275

islandzki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka islandzkiego.

Tabela 652. Rodzime identyfikatory CCSID dla języka islandzkiego na obsługiwanych platformach

Platforma	**Rodzime identyfikatory CCSID**
 IBM i  z/OS	871, 924, 1149
 AIX	819, 923, 5348
 Windows	850, 858, 861, 1252, 5348
 Linux	819, 923
Klient Apple	1275

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

871
Nie konwertuje na strony kodowe 858, 923, 924, 1275, 5348

924
Nie konwertuje na strony kodowe 858, 861, 871, 1051, 1149, 1252, 1275, 5348

1149
Nie konwertuje na strony kodowe 924, 1051, 1275

Windows

Windows

Strona kodowa:

861
Nie konwertuje na strony kodowe 1051, 1275

Języki wschodnioeuropejskie

Szczegóły dotyczące CCSID i konwersji CCSID dla języków wschodnioeuropejskich. Typowe języki korzystające z tych identyfikatorów CCSID to albański, chorwacki, czeski, węgierski, polski, rumuński, serbski, słowacki i słoweński.

Tabela 653. Rodzime identyfikatory CCSID dla języków wschodnioeuropejskich na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	870, 1153
Windows Windows	852, 1250, 5346, 9044
AIX AIX Linux Linux	912
Wschodnioeuropejski klient Apple	1282
rumuński klient Apple	1285
Chorwacki klient Apple	1284

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

870
Nie konwertuje na strony kodowe 1284, 1285

1153
Nie konwertuje na strony kodowe 1250, 1284, 1285

IBM i

IBM i

Strona kodowa:

870
Nie konwertuje na strony kodowe 1284, 1285, 5346, 9044

1153
Nie konwertuje na strony kodowe 1282, 1284, 1285, 5346, 9044

, Linux

Strona kodowa:

912
Nie konwertuje na strony kodowe 1284, 1285

Windows

Strona kodowa:

852
Nie konwertuje na strony kodowe 1284, 1285

1250
Nie konwertuje na strony kodowe 1284, 1285

9044
Nie konwertuje na strony kodowe 912, 1282, 1284, 1285

cyrylica

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla cyrylicy. Typowe języki używające tych identyfikatorów CCSID to: białoruski, bułgarski, macedoński, rosyjski i serbski.

Tabela 654. Rodzime identyfikatory CCSID dla cyrylicy na obsługiwanych platformach

Platforma	**Rodzime identyfikatory CCSID**
 z/OS	1025
 IBM i	880, 1025
 Windows	855, 866, 1131, 1251, 5347
 AIX  Linux	915
Klient Apple	1283

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

Strona kodowa:

880
Nie konwertuje na strony kodowe 855, 866, 878, 1131, 5347

1025
Nie konwertuje na strony kodowe 878, 5347

Windows

Strona kodowa:

855
Nie konwertuje na stronę kodową 1131

866
Nie konwertuje na stronę kodową 1131

1131
Nie konwertuje na strony kodowe 855, 866, 880, 1283

estoński

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka estońskiego.

Tabela 655. Rodzime identyfikatory CCSID dla języka estońskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	1122, 1157
Windows Windows	902, 922, 1257, 5353, 9449
AIX AIX Linux Linux	902, 922

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

1122
Nie konwertuje na strony kodowe 902, 1157, 9449

1157
Nie konwertuje na strony kodowe 922, 1122, 1257, 9449

IBM i

IBM i

Strona kodowa:

1122
Nie konwertuje na strony kodowe 902, 5353, 9449

1157
Nie konwertuje na strony kodowe 922, 5353, 9449

Linux

Linux

Strona kodowa:

902
Nie konwertuje na strony kodowe 922, 1122, 9449

922
Nie konwertuje na strony kodowe 902, 1157, 9449

Windows

Windows

Strona kodowa:

5353
Nie konwertuje na stronę kodową 9449

9449
Nie konwertuje na strony kodowe 902, 922, 1122, 1157, 1257, 5353

902
Nie konwertuje na strony kodowe 922, 1122, 9449

Łotewski i litewski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka łotewskiego i litewskiego.

Tabela 656. Rodzime identyfikatory CCSID dla języka łotewskiego i litewskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	1112, 1156
Windows Windows	901, 921, 1257, 5353, 9449
AIX AIX Linux Linux	901, 921

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

1112
Nie konwertuje na strony kodowe 901, 1156, 9449

1156
Nie konwertuje na strony kodowe 901, 1156, 9449

IBM i

IBM i

Strona kodowa:

1112
Nie konwertuje na stronę kodową 5353

1153
Nie konwertuje na strony kodowe 921, 5353, 9449

Linux

Linux

Strona kodowa:

902
Nie konwertuje na strony kodowe 921, 1112, 1257, 9449

921
Nie konwertuje na strony kodowe 901, 1156, 9449

Windows

Windows

Strona kodowa:

901
Nie konwertuje na strony kodowe 921, 1112, 1257, 9449

5355
Nie konwertuje na stronę kodową 9449

9449
Nie konwertuje na strony kodowe 901, 921, 1112, 1156, 1257

ukraiński

Szczegóły identyfikatorów CCSID i konwersji CCSID dla języka ukraińskiego.

Tabela 657. Rodzime identyfikatory CCSID dla języka ukraińskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	1123
Windows Windows	1124, 1125, 1251, 5347
AIX AIX Linux Linux	1124

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

1123
Nie konwertuje na stronę kodową 5347

Windows

Windows

Strona kodowa:

1125
Nie konwertuje na stronę kodową 1123

grecki

Szczegóły dotyczące CCSID i konwersji CCSID dla języka greckiego.

Tabela 658. Rodzime identyfikatory CCSID dla języka greckiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	875
Windows	869, 1253, 5349
AIX Linux NCR	813
Klient Apple	1280
Klient DOS	737

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID, czyli rodzimymi identyfikatorami CCSID innych platform z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

875
Nie konwertuje na stronę kodową 5349

Windows

Windows

Strona kodowa:

1253
Nie konwertuje na stronę kodową 737

5349
Nie konwertuje na stronę kodową 737

turecki

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka tureckiego.

Tabela 659. Rodzime identyfikatory CCSID dla języka tureckiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	1026
Windows	857, 1254, 5350
AIX Linux	920
Klient Apple	1281

Wszystkie platformy inne niż klienckie obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

1026
Nie konwertuje na stronę kodową 5350

hebrajski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka hebrajskiego.

Tabela 660. Rodzime identyfikatory CCSID dla języka hebrajskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
z/OS z/OS	424, 803, 4899, 12712
IBM i IBM i	424
AIX AIX	916, 9048
Windows Windows	1255, 5351
Linux Linux	916

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

424
Nie konwertuje na strony kodowe 867, 4899, 9048, 12712

803
Nie konwertuje na strony kodowe 867, 4899, 5351, 9048, 12712

4899
Nie konwertuje na strony kodowe 424, 803, 856, 862, 916, 1255

12712
Nie konwertuje na strony kodowe 424, 803, 856, 916, 1255

IBM i

IBM i

Strona kodowa:

424
Nie konwertuje na strony kodowe 803, 867, 4899, 5351, 9048, 12712

Strona kodowa 424 jest również konwertowany na i z CCSID 4952, który jest wariantem 856.

AIX

AIX

Strona kodowa:

916
Nie konwertuje na strony kodowe 867, 4899, 9048, 12712

9048
Nie konwertuje na strony kodowe 424, 803, 856, 862, 916, 1255

Windows

Windows

Strona kodowa:

1255
Nie konwertuje na strony kodowe 867, 4899, 9048, 12712

5351
Nie konwertuje na stronę kodową 803

arabski

Szczegóły dotyczące CCSID i konwersji CCSID dla języka arabskiego

Tabela 661. Rodzime identyfikatory CCSID dla języka arabskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	420
AIX AIX	1046, 1089
	1089 (patrz uwaga)
Windows Windows	720, 864, 1256, 5352
Linux Linux	1089

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

420
Nie konwertuje na stronę kodową 5352

Linux, Tru64

Linux

Strona kodowa:

1089
Nie konwertuje na stronę kodową 720

Windows

Windows

Strona kodowa:

720
Nie konwertuje na strony kodowe 1089, 5352

5352
Nie konwertuje na stronę kodową 720

farsi

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka farsi.

Tabela 662. Rodzime identyfikatory CCSID dla języka farsi na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	1097
AIX AIX Linux Linux Windows Windows	1098 (patrz uwaga)

Uwaga: Rodzimy identyfikator CCSID dla tych platform nie został ustandaryzowany i może ulec zmianie.

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform.

urdu

Szczegóły dotyczące CCSID i konwersji CCSID dla Urdu.

Tabela 663. Rodzime identyfikatory CCSID dla urdu na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	918
Windows Windows	868
AIX AIX Linux Linux	1006

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

918
Nie konwertuje na stronę kodową 1006

tajski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka tajskiego.

Tabela 664. Rodzime identyfikatory CCSID dla języka tajskiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	838
AIX Linux Windows	874 (patrz uwaga)

Uwaga: Rodzimy identyfikator CCSID dla tych platform nie został ustandaryzowany i może ulec zmianie.

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform.

laotański

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka laotańskiego.

Tabela 665. Rodzime identyfikatory CCSID dla języka laotańskiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	1132
AIX Linux Windows	1133

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID a rodzimymi identyfikatorami CCSID innych platform.

wietnamski

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka wietnamskiego.

Tabela 666. Rodzime identyfikatory CCSID dla języka wietnamskiego na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
IBM i z/OS	1130
Windows	1258, 5354
AIX Linux	1129

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

IBM i

IBM i

Strona kodowa:

1130
Nie konwertuje na strony kodowe 1129, 5354

japoński łaciński SBCS

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla japońskiego łacińskiego zestawu znaków SBCS.

Tabela 667. Rodzime identyfikatory CCSID dla japońskich łacińskich zestawów SBCS na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	1027
AIX AIX	932, 5050, 33722 (zob. uwaga 1)
Windows Windows	932, 943 (zob. uwaga 2)
Linux Linux	943, 5050

Uwaga:

1. AIX Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
2. Windows Produkt Windows NT używa strony kodowej 932, ale jest ona najlepiej reprezentowana przez identyfikator CCSID 943. Jednak nie wszystkie platformy IBM MQ obsługują ten identyfikator CCSID.

   W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można wprowadzić zmianę w pliku `../conv/table/ccsid.tbl` , która zmienia identyfikator CCSID używany na 943.

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

1027
Nie konwertuje na strony kodowe 932, 942, 943, 954, 5050, 33722

IBM i

IBM i

Strona kodowa:

1027
Nie konwertuje na stronę kodową 932

AIX

AIX

Strona kodowa:

932
Nie konwertuje na stronę kodową 1027

5050
Nie konwertuje na stronę kodową 1027

33722
Nie konwertuje na stronę kodową 1027

Linux

Linux

Strona kodowa:

943
Nie konwertuje na stronę kodową 1027

5050
Nie konwertuje na stronę kodową 1027

japoński Katakana SBCS

Szczegóły dotyczące CCSID i konwersji CCSID dla japońskiej Katakana SBCS.

Tabela 668. Rodzime identyfikatory CCSID dla japońskiej Katakana SBCS na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	290
AIX AIX	932, 5050, 33722 (zob. uwaga 1)
Windows Windows	932, 943 (zob. uwaga 2)
Linux Linux	943, 5050

Uwaga:

1. AIX Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
2. Windows Produkt Windows NT używa strony kodowej 932, ale jest ona najlepiej reprezentowana przez identyfikator CCSID 943. Jednak nie wszystkie platformy IBM MQ obsługują ten identyfikator CCSID.

   W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można wprowadzić zmianę w pliku `../conv/table/ccsid.tbl` , która zmienia identyfikator CCSID używany na 943.
3. Oprócz wcześniejszych konwersji, system IBM MQ obsługuje konwersję z CCSID 897 na CCSID 37, 273, 277, 278, 280, 284, 285, 290, 297, 437, 500, 819, 850, 1027 i 1252 na następujących platformach:
   - AIX AIX
   - Linux Linux

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

290
Nie konwertuje na strony kodowe 932, 943, 954, 5050, 33722

IBM i

IBM i

Strona kodowa:

290
Nie konwertuje na stronę kodową 932

AIX

AIX

Strona kodowa:

932
Nie konwertuje na strony kodowe 290, 897

5050
Nie konwertuje na strony kodowe 290, 897

33722
Nie konwertuje na strony kodowe 290, 897

Linux

Linux

Strona kodowa:

943
Nie konwertuje na strony kodowe 290, 897

5050
Nie konwertuje na strony kodowe 290, 897

Japoński Kanji/łaciński mieszany

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla japońskiego zestawu znaków Kanji/Latin Mixed.

Tabela 669. Rodzime identyfikatory CCSID dla języka japońskiego Kanji/Latin Mixed na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	1399, 5035 (zob. uwaga 1)
AIX AIX	932, 5050, 33722 (zob. uwaga 2)
Windows Windows	932, 943 (zob. uwaga 4)

Tabela 669. Rodzime identyfikatory CCSID dla języka japońskiego Kanji/Latin Mixed na obsługiwanych platformach (kontynuacja)	
Platforma	**Rodzime identyfikatory CCSID**
Linux	943, 5050

Uwaga:

1. IBM i z/OS 5035 jest identyfikatorem CCSID związanym ze stroną kodową 939
2. AIX Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
3. Windows Produkt Windows NT używa strony kodowej 932, ale jest ona najlepiej reprezentowana przez identyfikator CCSID 943. Jednak nie wszystkie platformy IBM MQ obsługują ten identyfikator CCSID.

   W systemie IBM MQ for Windows identyfikator CCSID 932 jest używany do reprezentowania strony kodowej 932, ale można wprowadzić zmianę w pliku `../conv/table/ccsid.tbl` , która zmienia identyfikator CCSID używany na 943.

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

1399
Nie konwertuje na strony kodowe 954, 5035, 5050, 33722

5035
Nie konwertuje na strony kodowe 954, 1399, 5050, 33722

IBM i

IBM i

Strona kodowa:

1399
Nie konwertuje na stronę kodową 5039

5035
Nie konwertuje na stronę kodową 5039

Japoński Kanji/Katakana Mieszany

Szczegóły dotyczące CCSID i konwersji CCSID dla japońskiego Kanji/Katakana Mieszanego.

Tabela 670. Rodzime identyfikatory CCSID dla japońskich znaków Kanji/Katakana mieszanych na obsługiwanych platformach	
Platforma	**Rodzime identyfikatory CCSID**
z/OS	1390, 5026 (patrz Uwaga “1” na stronie 994)
IBM i	5026 (patrz Uwaga “1” na stronie 994)

Tabela 670. Rodzime identyfikatory CCSID dla japońskich znaków Kanji/Katakana mieszanych na obsługiwanych platformach (kontynuacja)	
Platforma	**Rodzime identyfikatory CCSID**
AIX	932, 5050, 33722 (patrz Uwaga "2" na stronie 994)
Windows	932, 943 (patrz uwaga "3" na stronie 994)
Linux	943, 5050

Uwaga:

1. IBM i z/OS Tryb jednobajtowy identyfikatorów CCSID 1390 i 5026 w kodzie EBCDIC zawiera małe litery w różnych miejscach, w zależności od typowego lub niezmiennego układu dla podstawowego alfabetu łacińskiego. Dlatego należy zadbać o to, aby dane nie zostały utracone podczas konwersji danych komunikatu do innych identyfikatorów CCSID. Ponadto użycie tych identyfikatorów CCSID jako domyślnego identyfikatora CCSID menedżera kolejek może powodować problemy podczas komunikacji z innymi menedżerami kolejek. Na przykład nazwy kanałów zawierające małe litery mogą nie być poprawnie interpretowane w systemie zdalnym. 5026 jest identyfikatorem CCSID związanym ze stroną kodową 930. Identyfikator CCSID 5026 jest identyfikatorem CCSID zgłoszonym w systemie IBM i po wybraniu opcji Japanese Katakana (DBCS).
2. AIX Jednostki 5050 i 33722 są identyfikatorami CCSID powiązanymi z podstawową stroną kodową 954 w systemie AIX. Identyfikator CCSID zgłoszony przez system operacyjny to 33722.
3. Windows Produkt Windows NT używa strony kodowej 932, ale jest ona najlepiej reprezentowana przez identyfikator CCSID 943. Jednak nie wszystkie platformy IBM MQ obsługują ten identyfikator CCSID.

   W systemie IBM MQ for Windowsdo reprezentowania strony kodowej 932 używany jest identyfikator CCSID 932, ale można wprowadzić zmianę w pliku `../conv/table/ccsid.tbl` , która spowoduje zmianę identyfikatora CCSID używanego na 943.

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

1390
Nie konwertuje na strony kodowe 954, 5026, 5050, 33722

Małe litery nie są akceptowane.

5026
Nie konwertuje na strony kodowe 954, 1390, 5050, 33722

IBM i

IBM i

Strona kodowa:

5026
Nie konwertuje na strony kodowe 1390, 5039

koreański

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka koreańskiego.

Tabela 671. Rodzime identyfikatory CCSID dla języka koreańskiego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	933, 1364
AIX AIX Linux Linux	970
Windows Windows	949, 1363

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

933
Nie konwertuje na stronę kodową 970

1364
Nie konwertuje na stronę kodową 970

chiński uproszczony

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka chińskiego uproszczonego.

Tabela 672. Rodzime identyfikatory CCSID dla języka chińskiego uproszczonego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
z/OS z/OS	935, 1388
IBM i IBM i	935, 1388
AIX AIX	1383, 1386
Windows Windows	1381, 1386 (zob. uwaga 2)
Linux Linux	1383

Uwaga:

1. Windows System Windows używa strony kodowej 936, ale jest ona najlepiej reprezentowana przez identyfikator CCSID 1386. Jednak nie wszystkie platformy IBM MQ obsługują ten identyfikator CCSID.

   W systemie IBM MQ for Windows identyfikator CCSID 1381 jest używany do reprezentowania strony kodowej 936, ale można wprowadzić zmianę w pliku `../conv/table/ccsid.tbl` , która zmienia identyfikator CCSID używany na 1386.
2. IBM MQ obsługuje chiński standard GB18030 .

Windows Linux z/OS W systemach z/OS, Windows i Linuxobsługa konwersji jest zapewniana między Unicode (UTF-8 i UTF-16) i CCSID 1388 (EBCDIC z rozszerzeniami GB18030), Unicode (UTF-8 i UTF-16) i CCSID 5488 (GB18030) oraz między CCSID 1388 i CCSID 5488.

Uwaga: Identyfikator CCSID musi mieć wartość 5488, aby można było używać GB18030 znaków. Nie można jednak ustawić identyfikatora CCSID w menedżerze kolejek utworzonym za pomocą konsoli IBM MQ Explorer lub konsoli IBM MQ Console. Zamiast tego należy utworzyć menedżer kolejek przy użyciu interfejsu CLI z identyfikatorem CCSID 5488 lub użyć wiersza komend interfejsu CLI, aby zmienić identyfikator CCSID po utworzeniu menedżera kolejek.

IBM i W systemie IBM isystem operacyjny zapewnia obsługę konwersji między kodowaniem Unicode (UTF-8 i UTF-16) a kodowaniem CCSID 1388 (EBCDIC z rozszerzeniami GB18030).

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

935
Nie konwertuje na stronę kodową 1383

1388
Nie konwertuje na stronę kodową 1383

chiński tradycyjny

Szczegóły dotyczące identyfikatorów CCSID i konwersji CCSID dla języka chińskiego tradycyjnego.

Tabela 673. Rodzime identyfikatory CCSID dla języka chińskiego tradycyjnego na obsługiwanych platformach

Platforma	Rodzime identyfikatory CCSID
IBM i IBM i z/OS z/OS	937
Windows Windows	950
AIX AIX Linux Linux	950, 964

Wszystkie platformy obsługują konwersję między rodzimymi identyfikatorami CCSID i rodzimymi identyfikatorami CCSID innych platform, z następującymi wyjątkami.

z/OS

z/OS

Strona kodowa:

937
Nie konwertuje na stronę kodową 964

1388
Nie konwertuje na stronę kodową 1383

Linux

Linux

Strona kodowa:

964
Nie konwertuje na stronę kodową 938

z/OS Obsługa konwersji z/OS

Lista obsługiwanych konwersji CCSID.

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS

CCSID	Konwertuje do i z CCSID
37	256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-11243, 1047, 1051 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709
256	37, 273, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 819, 833, 836, 838, 850, 852, 857, 860-866, 869-871, 875, 880, 905, 1025-1027, 1112, 1122, 1200, 1208, 1251-1252, 1275, 4386, 4929, 4932, 4934, 4946, 4948, 4953, 4960, 71
259	437, 808, 850-852, 855-858, 860-865, 867, 869, 872, 874, 899, 901-902, 915, 1098, 1161-1162, 1200, 1208, 1250-1258, 4946, 4948, 4951-4953, 4960, 4970, 5346, 5348, 9044, 9049, 9056, 9061, 9066, 13488, 17248, 17584
273	37, 256, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 920, 923-924, 1009, 1025-1027, 1040-1049, 104951, 1088, 1141, 1112, 112, 112, 112
274	500, 1047
275	37, 437, 500, 819, 850, 1047, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
277	37, 256, 273, 278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 104943, 1047, 1051, 1141088, 1100, 1111, 12, 12
278	37, 256, 273, 277, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 104943, 1047, 1051, 1141088, 1100, 1112, 1212
280	37, 256, 273, 277-278, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 104943, 1051, 1141088, 1100, 1111, 12
281	1047
282	500, 1047, 1200, 1208, 13488, 17584

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
284	37, 256, 273, 277-278, 280, 285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 104943, 1047, 1051, 1141088, 1100, 1112, 1112
285	37, 256, 273, 277-278, 280, 284, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 86971, 874-875, 880, 897, 903, 912, 916, 920, 923-924, 1025-1027, 1040-1043, 1051, 1088, 1100, 1112, 1122, 1120
290	37, 256, 273, 277-278, 280, 284-285, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4960, 4992
293	1200, 1208, 13488, 17584
297	37, 256, 273, 277-278, 280, 284-285, 290, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-865, 869-871, 874-875, 897, 903, 912, 916, 920, 923-924, 1009, 1025-1027, 104943, 1047, 1051, 1141088, 1100, 1112,
300	301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
301	300, 941, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
367	37, 256, 273, 277-278, 280, 284, 290, 297, 500, 819, 833, 836, 850, 871, 875, 1009, 1026-1027, 1041, 1088, 1115, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4971, 5123, 5211, 8229, 8482, 9025, 13121, 13488, 17584, 25617, 25664, 28709
420	37, 256, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
423	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 737, 775, 813, 819, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953,
424	37, 256, 420, 437, 500, 737, 775, 803, 819, 836, 850, 852, 856-857, 860-865, 916, 1112, 1122, 1200, 1208, 1252, 1255, 4932, 4946, 4948, 4952-4953, 4960, 5012, 5351, 8229, 8612, 9044, 9049, 9056, 13488, 16804, 17248, 17584, 28709
437	37, 256, 259, 273, 275, 277-278, 280, 284-285, 297, 420, 423-424, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-863, 865-866, 869-871, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1040-1043, 1047, 1051, 25473, 25479, 25617, 25619, 28709
500	37, 256, 273-275, 277-278, 280, 282, 284-285, 290, 297, 367, 420, 423-424, 437, 737, 775, 813, 819, 833, 836, 838, 850-852, 855-858, 860-866, 869-871, 874-875, 880, 891, 895, 897, 903-905, 912, 914-916, 920-4924, 1004, 1009-1021, 1025-1027, 5348, 8229, 8482, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 9238, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664, 28709

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
720	37, 420, 864, 1200, 1208, 1256, 4960, 8229, 8612, 9056, 13488, 16804, 17248, 17584, 28709
737	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 833, 836, 838, 850, 869-871, 875, 880, 905, 1025-1027, 1097, 1200, 1208, 1252-1253, 1280, 4386, 4909, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 489030, 9061, 131,
775	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 833, 836, 838, 850, 870-871, 875, 880, 905, 1025-1027, 1097, 1112, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 8229, 8482, 8612, 9025, 9030, 13121, 13488, 16804
803	424, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1255, 4946, 4952, 5012, 13488, 17584
806	1200, 1208, 13488, 17584
808	259, 858-859, 872, 923-924, 1140, 1148, 1153-1154, 1200, 1208, 5347, 5348, 13488, 17584
813	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 593, 549, 499,
819	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 803, 813, 833, 836, 838, 850, 852, 855, 857-858, 860-861, 863-866, 869-875, 880, 897, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1041-1043, 1047, 1047, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
833	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 821329, 82
834	926, 951, 1200, 1208, 1362, 4930, 9026, 13488, 17584
835	927, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
836	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 5210
837	928, 1200, 1208, 1380, 1385, 4933, 13488, 17584
838	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 4953, 4960, 0612, 4970
848	924, 1148, 1158, 1200, 1208, 5347, 13488, 17584
849	924, 1148, 1154, 1200, 1208, 5347, 13488, 17584

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
850	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 855-858, 860-866, 86971, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1143, 1047, 1047 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 25619, 25664, 28709
851	259, 423, 500, 875, 1200, 1208, 4971, 13488, 17584
852	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 1088, 1097, 1200, 1208, 1250, 1252, 1282
855	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 5346, 5347, 8229, 8482, 9025,
856	259, 273, 424, 500, 803, 850, 862, 916, 1200, 1208, 1255, 4946, 4952, 5012, 5351, 13488, 17584
857	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1200, 1208, 1252, 1254, 481, 4386
858	37, 259, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 860-861, 865, 871-872, 901-902, 923-924, 1047, 1051, 1140-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
859	808, 872, 901-902, 1153-1157, 1160-1162, 1164, 1200, 1208, 13488, 17584
860	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 861, 863, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025, 1027, 1-1043, 1097, 1140, 1145-1146, 1148, 1200
861	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857-858, 860, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 923-924, 1025-1027, 1041-1043, 1097, 498, 114499, 1200, 1208, 1252, 4386
862	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 803, 833, 838, 850, 856, 870-871, 875, 880, 905, 916, 1025-1027, 1097, 1200, 1208, 1252, 1255, 4386, 4929, 4934, 4946, 4952, 4971, 5012, 5123, 5351, 8229, 8482, 8612, 9025, 139030,
863	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 838, 850, 852, 857, 860-861, 865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1041-1043, 1051, 1097, 114499, 1200, 1208, 1252, 1275, 475, 4386
864	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 5142, 485352, 8229, 82

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
865	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 819, 833, 838, 850, 858, 860, 863, 870-871, 875, 880, 905, 923-924, 1025-1027, 1097, 1142-1143, 1148, 1200, 1208, 1252, 4386, 4929, 4934, 4946, 4971, 5123, 5348, 8229, 488902, 138612, 4825,
866	37, 256, 437, 500, 819, 850, 855, 870, 878, 880, 915, 1025, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
867	259, 1153-1155, 1160, 1200, 1208, 4899, 5351, 9048, 12712, 13488, 17584
868	918, 1006, 1200, 1208, 13488, 17584
869	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 870-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4970-4971, 12
870	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869, 871, 874-875, 880, 897, 903, 912, 915-916, 920, 1009, 1025-1027, 1040-1043, 1047, 4988, 1112, 1122, 1200, 1208, 1252.
871	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 858, 860-865, 869, 870, 874-75, 880, 897, 903, 916, 920, 923-924, 1009, 1025-1027, 1040-4943, 1047, 11451, 1088, 1112,
872	259, 808, 858-859, 923-924, 1140-1149, 1153-1155, 1200, 1208, 5347, 5348, 13488, 17584
874	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 492553, 4970-4971, 5012, 5123, 068229, 9030,
875	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 4909,
878	855, 866, 880, 915, 1025, 1131, 1200, 1208, 1251, 1283, 4951, 5347, 13488, 17584
880	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 878, 897, 903, 912, 915 -916, 920, 1009, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1251-1252, 1283, 4909, 4934,
891	500, 833, 1088, 1200, 1208, 4929, 9025, 13121, 13488, 17584, 25664
895	290, 500, 1027, 1041, 1200, 1208, 4386, 5123, 8482, 13488, 17584, 25617
896	290, 1027, 1041, 1200, 1208, 4386, 4992, 5123, 8482, 13488, 17584, 25617

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
897	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4386, 4909, 4934, 4946, 4948, 4953, 4970-4971, 4812, 5123, 8229, 830,
899	259
901	259, 858-859, 902, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
902	259, 858-859, 901, 923-924, 1140, 1148, 1156-1157, 1200, 1208, 5348, 5353, 13488, 17584
903	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1200, 1208, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123,
904	37, 500, 1114, 1200, 1208, 5210, 8229, 13488, 17584, 25480, 28709
905	37, 256, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 920, 1026, 1112, 1122, 1200, 1208, 1252, 1254, 1281, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
912	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 916, 920, 1025-1027, 1041-1043, 1047, 1200, 1208, 1250, 1252, 1282, 4909, 4934, 4946, 4948, 4951, 4953, 4953, 49494970.
914	37, 437, 500, 819, 850, 1200, 1208, 1252, 1257, 4946, 8229, 13488, 17584, 28709
915	37, 259, 437, 500, 819, 850, 855, 866, 870, 878, 880, 1025, 1131, 1200, 1208, 1251-1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
916	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 46, 4948, 4952-4953, 4970-4971, 5012, 5123
918	864, 868, 1006, 1200, 1208, 4960, 9056, 13488, 17248, 17584
920	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 1025-1026, 1200, 1208, 1252, 1254, 1281, 4909, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5350, 068229, 9030,
921	37, 437, 500, 819, 850, 922, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
922	37, 437, 500, 819, 850, 921, 1112, 1122, 1200, 1208, 1252, 1257, 4946, 5353, 8229, 13488, 17584, 28709
923	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 865, 871-872, 901-902, 924, 1047, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
924	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 865, 871-872, 901-902, 923, 1047, 1051, 1140-1149, 1153-1157, 1160-1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
926	834, 951, 9026
927	835, 947, 1200, 1208, 4931, 9027, 13488, 17584, 21427
928	837, 1200, 1208, 1380, 13488, 17584
930	931-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
931	930, 932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
932	930-931, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
933	934, 944, 949, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
934	933, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25510, 25525, 29621, 33717, 37813
935	936, 946, 1200, 1208, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
936	935, 946, 1381, 5031, 5477, 5484, 9127, 13223, 25512
937	938, 948, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
938	937, 950, 1370, 5033, 5046, 9142, 25514
939	930-932, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
941	300-301, 1200, 1208, 1351, 4396, 8492, 13488, 16684, 17584
942	930-932, 939, 943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
943	930-932, 939, 942, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
944	933, 949, 1200, 1208, 5029, 5045, 5460, 9125, 13221, 13488, 17317, 17584, 25520, 25525, 29616, 29621, 33717, 37813
946	935-936, 1200, 1208, 5031, 5484, 9127, 13223, 13488, 17584, 25512
947	835, 927, 1200, 1208, 4931, 9027, 13488, 17584, 21427

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
948	937, 950, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25524, 29620
949	933-934, 944, 1200, 1208, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25510, 25520, 25525, 29616, 29621, 33717, 37813
950	937-938, 948, 1200, 1208, 1370, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
951	834, 926, 1200, 1208, 1362, 4930, 9026, 13488, 17584
1004	500, 819, 850, 1200, 1208, 4946, 13488, 17584
1006	868, 918, 1200, 1208, 13488, 17584
1008	420, 864, 1200, 1208, 4960, 5104, 8612, 9056, 13488, 16804, 17248, 17584
1009	37, 273, 277-278, 280, 284, 290, 297, 367, 423, 500, 833, 836, 870-871, 875, 880, 1025-1026, 1200, 1208, 4386, 4929, 4932, 4971, 8229, 8482, 9025, 13121, 13488, 17584, 28709
1010	500, 1200, 1208, 13488, 17584
1011	500, 1200, 1208, 13488, 17584
1012	500, 1200, 1208, 13488, 17584
1013	500, 1140, 1200, 1208, 13488, 17584
1014	500, 1200, 1208, 13488, 17584
1015	500, 1200, 1208, 13488, 17584
1016	500, 1200, 1208, 13488, 17584
1017	500, 1200, 1208, 13488, 17584
1018	500, 1200, 1208, 13488, 17584
1019	500, 1200, 1208, 13488, 17584
1020	500
1021	500
1023	500
1025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-866, 869-871, 874-875, 880, 897, 903, 912, 915 -916, 920, 1009, 1026-1027, 1040-43, 1051, 1088, 1112, 1122, 1131
1026	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1009, 1025, 1027, 104943, 1088, 1112, 1122, 1200, 1049
1027	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-865, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1026, 1040-1043, 1047, 1088, 114912, 1122, 1139, 1200, 1208, 1252

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
1040	37, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 833, 836, 850, 852, 855, 857, 870-871, 1025-1027, 1041-1043, 1088, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617, 25619, 25664, 289
1041	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040, 1042-1043, 1088, 1200, 1208, 1252, 4386, 4909, 4929, 4932, 4934
1042	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1043, 1088, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4953, 4946, 4948
1043	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040, 1041, 1042, 1088, 1114, 1200, 1208, 4386, 4909, 4929, 4932, 4934, 4946,
1046	420, 500, 864, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1047	37, 273-275, 277-278, 280, 281, 282, 284-285, 290, 297, 437, 500, 819, 850, 852, 858, 870-871, 875, 912, 923-924, 1026-1027, 1140-1149, 1200, 1208, 1252, 1254, 4946, 4948, 5123, 8229, 8482, 13488, 17584, 28709
1051	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1025, 1097, 1140-1149, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1088	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1126, 1200, 1208, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 13488, 17584, 25617,
1089	420, 500, 819, 850, 864, 1046, 1127, 1200, 1208, 1256, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1097	37, 437, 500, 737, 775, 819, 850, 852, 857, 860-865, 1051, 1098, 1112, 1122, 1200, 1208, 1252, 4946, 4948, 4953, 4960, 8229, 9044, 9049, 9056, 13488, 17248, 17584, 28709
1098	259, 420, 437, 819, 850, 1097, 1200, 1208, 1252, 4946, 8612, 13488, 16804, 17584
1100	37, 273, 277-278, 280, 284-285, 297, 500, 850, 4946, 8229, 28709
1101	500
1102	500
1103	500
1104	500
1105	500
1106	500

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
1107	500
1112	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1122, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 139030,
1114	37, 437, 500, 819, 836, 850, 904, 1043, 1115, 1200, 1208, 4932, 4946, 5210-5211, 8229, 13488, 17584, 25480, 25619, 28709
1115	37, 367, 437, 500, 836, 903, 1114, 1200, 1208, 4932, 5210-5211, 8229, 13488, 17584, 25479, 28709
1122	37, 256, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 775, 819, 833, 836, 838, 850, 870-871, 875, 880, 905, 921-922, 1025-1027, 1097, 1112, 1200, 1208, 1252, 1257, 4386, 4929, 4932, 4934, 4946, 4971, 5123, 5353, 8229, 8482, 8612, 9025, 139030,
1123	819, 1124-1125, 1148, 1200, 1208, 1251-1252, 1283, 5347, 13488, 17584
1124	37, 500, 1123, 1125, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1125	500, 1123, 1124, 1200, 1208, 1251, 1283, 5347, 13488, 17584
1126	37, 367, 437, 500, 819, 833, 850, 1088, 1200, 1208, 1252, 4929, 4946, 8229, 9025, 13121, 13488, 17584, 25664, 28709
1127	420, 864, 1046, 1089, 1256, 4960, 5142, 8612, 9056, 9238, 16804, 17248
1129	500, 1130, 1200, 1208, 1258, 5354, 13488, 17584
1130	37, 500, 819, 850, 1129, 1200, 1208, 1252, 1258, 4946, 5354, 8229, 13488, 17584, 28709
1131	37, 500, 878, 915, 1025, 1200, 1208, 1251, 1283, 5347, 8229, 13488, 17584, 28709
1132	37, 500, 819, 850, 1133, 1200, 1208, 1252, 4946, 8229, 13488, 17584, 28709
1133	500, 1132, 1200, 1208, 13488, 17584
1137	37, 500, 819, 1200, 1208, 8229, 13488, 17584, 28709
1139	290, 1027, 4386, 5123, 8482
1140	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860, 863, 871-872, 901-902, 923-924, 1013, 1047, 1051, 1141-1149, 1153-1157, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1141	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140, 1142-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1142	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1141, 1143-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
1143	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 865, 871-872, 923-924, 1047, 1051, 1140-1142, 1144-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1144	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1143, 1145-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1145	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1144, 1146-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1146	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 860, 863, 871-872, 923-924, 1047, 1051, 1140-1145, 1147-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1147	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871-872, 923-924, 1047, 1051, 1140-1146, 1148-1149, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1148	37, 273, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 848-850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1047, 1051, 1123, 1140-1147, 1149, 1153-1164, 1200, 1208, 1252, 1275, 4899, 4946, 5348, 5349, 8229, 12712, 13488, 17584, 28709
1149	37, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 861, 863, 871-872, 923-924, 1047, 1051, 1140-1148, 1153-1157, 1160-1162, 1200, 1208, 1252, 1275, 4946, 5348, 8229, 13488, 17584, 28709
1153	808, 858-859, 867, 872, 923-924, 1140-1149, 1154-1157, 1160-1162, 1200, 1208, 5348, 9044, 13488, 17584
1154	808, 849, 858-859, 867, 872, 923-924, 1140-1149, 1153, 1155-1157, 1160-1162, 1200, 1208, 5347, 5348, 13488, 17584
1155	858-859, 867, 872, 923-924, 1140-1149, 1153-1154, 1156-1157, 1160-1162, 1200, 1208, 5348, 5350, 9049, 13488, 17584
1156	858-859, 901-902, 923-924, 1140-1149, 1153-1155, 1157, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1157	858-859, 901-902, 923-924, 1140-1149, 1153-1156, 1160, 1200, 1208, 5348, 5353, 12712, 13488, 17584
1158	848, 923, 1148, 1200, 1208, 5347, 5348, 13488, 17584
1159	1148, 1200, 1208, 13488, 17584
1160	858-859, 867, 923-924, 1140-1149, 1153-1157, 1161-1162, 1200, 1208, 5348, 13488, 17584
1161	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1162	259, 858-859, 923-924, 1140-1149, 1153-1155, 1160, 5348, 17584
1163	924, 1148, 1164, 5354, 17584

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
1164	858-859, 923-924, 1140, 1148, 1163, 1200, 1208, 5348, 5354, 13488, 17584
1166	1200,1208,13488,17584
1200	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 90584, 9061, 9066, 9238, 13238, 12,
1208	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 920-924, 927 1362-1364, 1370-1371, 1374-1379, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5026, 5035, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9044, 9048-9049, 9056, 9061, 9066,
1250	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1252, 1282, 4946, 4948, 4951, 5346, 8229, 9044, 13488, 17584, 28709
1251	37, 256, 259, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1252, 1283, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1252	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855, 857-858, 860-866, 869-871, 874-875, 897, 903, 905, 912, 914-916, 920-924, 1025-1027, 1041, 1047, 1051, 1051 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25617, 28709
1253	37, 259, 423, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1280, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1254	37, 259, 500, 819, 850, 857, 869, 905, 920, 1026, 1047, 1200, 1208, 1252, 1281, 4946, 4953, 5350, 8229, 9049, 9061, 13488, 17584, 28709
1255	37, 259, 424, 500, 803, 819, 850, 856, 862, 916, 1200, 1208, 1252, 1281, 4946, 4952, 5012, 5351, 8229, 13488, 17584, 28709
1256	259, 420, 500, 720, 850, 864, 1046, 1089, 1127, 1200, 1208, 4946, 4960, 5142, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
1257	37, 259, 437, 500, 775, 819, 850, 914, 921-922, 1112, 1122, 1200, 1208, 1252, 4946, 5353, 8229, 13488, 17584, 28709
1258	37, 259, 500, 819, 1129-1130, 1200, 1208, 5354, 8229, 13488, 17584, 28709
1275	37, 256, 273, 277-278, 280, 284-285, 297, 437, 500, 819, 850, 858, 863, 871, 923-924, 1051, 1140-1149, 1200, 1208, 1252, 4946, 5348, 8229, 13488, 17584, 28709
1276	1200, 1208, 13488, 17584

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
1277	1200, 1208, 13488, 17584
1280	37, 423, 437, 500, 737, 813, 819, 850, 869, 875, 1200, 1208, 1252-1253, 4909, 4946, 4971, 5349, 8229, 9061, 13488, 17584, 28709
1281	37, 437, 500, 819, 850, 857, 905, 920, 1026, 1200, 1208, 1252, 1254-1255, 4946, 4953, 5350, 8229, 9049, 13488, 17584, 28709
1282	500, 852, 870, 912, 1200, 1208, 1250, 4948, 5346, 9044, 13488, 17584
1283	37, 437, 500, 819, 850, 855, 866, 878, 880, 915, 1025, 1123-1125, 1131, 1200, 1208, 1251-1252, 4946, 4951, 5347, 8229, 13488, 17584, 28709
1284	1200, 1208, 13488, 17584
1285	1200, 1208, 13488, 17584
1351	300-301, 941, 1200, 1208, 4396, 8492, 13488, 16684, 17584
1362	834, 951, 1200, 1208, 4930, 9026, 13488, 17584
1363	933, 949, 1200, 1208, 1364, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1364	933, 949, 1200, 1208, 1363, 5029, 5045, 5460, 9125, 9555, 13221, 13488, 13651, 17317, 17584, 25525, 29621, 33717, 37813
1370	937-938, 948, 950, 1200, 1208, 1371, 5033, 5046, 9142, 13488, 17584, 25514, 25524, 29620
1371	1200, 1208, 1370, 13488, 17584
1374	1200, 1208
1375	1200, 1208
1376	1200, 1208
1377	1200, 1208
1378	1200, 1208
1379	1200, 1208
1380	837, 928, 1200, 1208, 1385, 4933, 13488, 17584
1381	935-936, 1200, 1208, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584, 25512
1385	837, 1200, 1208, 1380, 4933, 13488, 17584
1386	935, 1200, 1208, 1381, 1388, 5031, 5477, 5482, 5484, 9127, 13223, 13488, 17584
1388	935, 1200, 1208, 1381, 1386, 5031, 5477, 5482, 5484, 5488, 9127, 13223, 13488, 17584
1390	930-932, 939, 942-943, 1200, 1208, 1399, 5026, 5028, 5035, 5038-5039, 5055, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
1399	930-932, 939, 942-943, 1200, 1208, 1390, 5026, 5028, 5035, 5038-5039, 5050, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
4386	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1139, 1252, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 4992, 5123, 89029,
4396	300-301, 941, 1351, 8492, 16684
4899	867, 1148, 1200, 1208, 5351, 9048, 12712, 13488, 17584
4909	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1253, 1280, 4934, 4946, 4948, 4953, 4970-4971, 5012, 065123, 5349,
4929	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482, 9025,
4930	834, 951, 1200, 1208, 1362, 9026, 13488, 17584
4931	835, 927, 947, 9027, 21427
4932	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 424, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 903, 1009, 1025-1027, 1040-1043, 1088, 1112, 1114-1115, 1122, 1252, 4386, 4929, 4946, 4948, 4951, 4953, 4971, 5123, 5210-5211, 8229,
4933	837, 1200, 1208, 1380, 1385, 13488, 17584
4934	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1252, 4909, 4946, 4948, 4953, 4960, 494971, 5012, 5123
4946	37, 256, 259, 273, 275, 277-278, 280, 284-285, 290, 367, 420, 423-424, 437, 500, 737, 775, 803, 813, 819, 833, 836, 838, 850, 852, 855-858, 860-866, 86971, 874-875, 880, 897, 903, 905, 912, 914-916, 920-924, 1004, 1025-1027, 1040-1043, 1041, 9066, 13121, 16804, 17248, 25473, 25479, 25617, 25619, 25664, 28709
4948	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 4988, 1097, 1200, 1208, 1250, 1252
4951	37, 259, 273, 277-278, 280, 284-285, 290, 297, 437, 500, 819, 833, 836, 850, 852, 855, 857, 866, 870-871, 878, 880, 912, 915, 1025-1027, 1040-1043, 1088, 1200, 1208, 1250-1252, 1283, 4386, 4929, 4932, 4946, 4948, 4953, 5123, 5346, 5347, 8229, 8482, 9025,
4952	259, 273, 424, 500, 803, 850, 856, 862, 916, 1200, 1208, 1255, 4946, 5012, 5351, 13488, 17584
4953	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1252, 1254, 1281, 4386, 4386,

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
4960	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4971, 5104, 5123, 5142, 5352, 8212, 8482
4970	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4909, 4934, 4946, 4948, 4953, 4971, 5012, 5123, 8229, 9030, 069044, 9049,
4971	37, 256, 273, 277-278, 280, 284-285, 297, 367, 423, 437, 500, 737, 775, 813, 819, 836, 838, 850-852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1009, 1025-1027, 1041-1043, 1047, 1088, 1112, 1122, 1200, 1208, 1252-1253, 1280, 490
4992	290, 896, 1027, 1041, 4386, 5123, 8482, 25617
5012	37, 273, 277-278, 280, 284-285, 297, 423-424, 437, 500, 803, 813, 819, 838, 850, 852, 856-857, 860-863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 1255, 4909, 4934, 4946, 4948, 4952-4953, 4970, 570, 5123
5026	930-932, 939, 942-943, 1390, 1399, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5028	930-932, 939, 942-943, 1390, 1399, 5026, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5029	933-934, 944, 949, 1363-1364, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5031	935-936, 946, 1381, 1386, 1388, 5477, 5482, 5484, 9127, 13223, 25512
5033	937-938, 948, 950, 1370, 5046, 9142, 25514, 25524, 29620
5035	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5038-5039, 9122, 9124, 9131, 9135, 1208, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5038	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
5039	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038, 9122, 9124, 9131, 9135, 13218-13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
5045	933-934, 944, 949, 1363-1364, 5029, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5046	937-938, 948, 950, 1370, 5033, 9142, 25514, 25524, 29620
5104	420, 864, 1008, 1200, 1208, 4960, 8612, 9056, 13488, 16804, 17248, 17584
5123	290, 367, 423, 437, 819, 1027, 1041, 1047, 1140-1149, 1156, 1157, 1160, 1200, 1208, 1252, 4948, 5348, 8482, 13488

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
5142	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5352, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5210	37, 437, 500, 819, 836, 850, 904, 1043, 1114-1115, 1200, 1208, 4932, 4946, 5211, 8229, 13488, 17584, 25480, 25619, 28709
5211	37, 367, 437, 500, 836, 903, 1114-1115, 4932, 5210, 8229, 25479, 28709
5346	37, 259, 273, 500, 819, 850, 852, 855, 870, 912, 1200, 1208, 1250, 1252, 1282, 4946, 4948, 4951, 8229, 9044, 13488, 17584, 28709
5347	808, 848-849, 855, 866, 872, 878, 880, 915, 1025, 1123-1125, 1131, 1154, 1158, 1200, 1208, 1251, 1283, 4951, 13488, 17584
5348	37, 259, 273, 275, 277-278, 280, 284-285, 297, 437, 500, 808, 819, 850, 858, 860-861, 863, 865, 871-872, 901-902, 923-924, 1051, 1140-1149, 1153-1158, 1160-1162, 1164, 1200, 1208, 1252, 1275, 4946, 8229, 13488, 17584, 28709
5349	813, 869, 875, 1148, 1200, 1208, 1253, 1280, 4909, 4971, 9061, 13488, 17584
5350	857, 920, 1026, 1155, 1200, 1208, 1254, 1281, 4953, 9049, 13488, 17584
5351	424, 856, 862, 867, 916, 1200, 1208, 1255, 4899, 4952, 5012, 9048, 12712, 13488, 17584
5352	420, 864, 1046, 1089, 1200, 1208, 1256, 4960, 5142, 8612, 9056, 9238, 13488, 16804, 17248, 17584
5353	901-902, 921-922, 1112, 1122, 1156-1157, 1200, 1208, 1257, 13488, 17584
5354	1129-1130, 1163, 1164, 1200, 1208, 1258, 13488, 17584
5460	933-934, 944, 949, 1363-1364, 5029, 5045, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
5477	935-936, 1381, 1386, 1388, 5031, 5482, 5484, 9127, 13223, 25512
5482	935, 1381, 1386, 1388, 5031, 5477, 5484, 9127, 13223
5484	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 9127, 13223, 25512
5488	1388
8229	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 858, 860-866, 86971, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 1040-1043, 1047, 1047, 947, 947 9044, 9049, 9056, 9061, 9066, 13121, 16804, 17248, 25473, 25479, 25480, 25617, 25619, 25664, 28709
8482	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 895-897, 1009, 1025-1027, 1040-1043, 1047, 1088, 1112, 1122, 1139, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 560,
8492	300-301, 941, 1351, 4396, 16684

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
8612	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 9044, 9049, 9056, 9238, 13488, 16804, 17248, 17584, 28709
9025	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229, 8482,
9026	834, 926, 951, 1362, 4930
9027	835, 927, 947, 1200, 1208, 4931, 13488, 17584, 21427
9030	37, 256, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 775, 813, 819, 838, 850, 852, 857, 860-865, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1112, 1122, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 248, 4953, 4960, 570
9044	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 912, 916, 920, 1025-1027, 1040-1043, 1047, 4988, 1097, 1153, 1200, 1208, 1250
9048	867, 1200, 1208, 4899, 5351, 12712, 13488, 17584
9049	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 905, 916, 920, 1025-1027, 1040-1043, 1088, 1097, 1155, 1200, 1208, 1252, 1254
9056	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 4946, 4960, 4971, 5104, 5123, 248, 485142, 5352, 8229,
9061	37, 256, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 737, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252-1254, 1280, 4909, 4934, 4946, 4948, 4953, 4950-4971,
9066	37, 259, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1200, 1208, 1252, 4909, 4934, 4946, 4948, 254953, 4970-4971, 5012, 5123, 8229,
9122	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9124	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9125	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
9127	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 13223, 25512

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
9131	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9135	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
9142	937-938, 948, 950, 1370, 5033, 5046, 25514, 25524, 29620
9238	420, 500, 864, 1046, 1089, 1127, 1200, 1208, 1256, 4960, 5142, 5352, 8612, 9056, 13488, 16804, 17248, 17584
9555	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 13221, 13651, 17317, 25525, 29621, 33717, 37813
12712	862, 867, 1148, 1156-1157, 1200, 1208, 4899, 5351, 9048, 13488, 17584
13121	37, 256, 273, 277-278, 280, 284-285, 290, 297, 367, 437, 500, 737, 775, 819, 833, 836, 850, 852, 855, 857, 860-865, 870-871, 891, 1009, 1025-1027, 1040-1043, 1088, 1112, 1122, 1126, 1200, 1208, 1252, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4960, 5123, 8229
13218	930-932, 939, 942-943, 1200, 1208, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13219, 13231, 13488, 17314, 17584, 25508, 25518, 29614, 33698-33700, 37796
13219	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218, 13231, 17314, 25508, 25518, 29614, 33698-33700, 37796
13221	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717, 37813
13223	935-936, 946, 1381, 1386, 1388, 5031, 5477, 5482, 5484, 9127, 25512
13231	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 17314, 25508, 25518, 29614, 33698-33700, 37796
13488	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 9238, 248, 12712, 12712, 13121, 13121, 17218, 13121, 13121, 13121
13651	933, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 17317, 25525, 29621, 33717, 37813
16684	300-301, 941, 1200, 1208, 1351, 4396, 8492, 13488, 17584
16804	37, 256, 420, 424, 437, 500, 720, 737, 775, 819, 850, 852, 857, 860-865, 1008, 1046, 1089, 1098, 1112, 1122, 1127, 1200, 1208, 1252, 1256, 4946, 4948, 4953, 4960, 5104, 5142, 5352, 8229, 8612, 9044, 9049, 9056, 9238, 13488, 17248, 17584, 28709

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
17248	37, 256, 259, 273, 277-278, 280, 284-285, 290, 297, 420, 423-424, 500, 720, 819, 833, 838, 850, 864, 870-871, 875, 880, 905, 918, 1008, 1025-1027, 1046, 1089, 1097, 1127, 1200, 1208, 1252, 1256, 4386, 4929, 4934, 498646, 4960, 4971, 5104, 5123, 485142, 5352, 8229,
17314	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 25508, 25518, 29614, 33698-33700, 37796
17317	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 25510, 25520, 25525, 29616, 29621, 33717, 37813
17584	37, 256, 259, 273, 275, 277-278, 280, 282, 284-285, 290, 293, 297, 300-301, 367, 420, 423-424, 437, 500, 720, 737, 775, 803, 806, 808, 813, 819, 833-838, 848-852, 855-872, 874-875, 878, 880, 891, 895-897, 901-905, 912, 914-916, 918, 920-924, 927 1362-1364, 1370-1371, 1380-1381, 1385-1386, 1388, 1390, 1399, 4899, 4909, 4930, 4933, 4948, 4951-4952, 4960, 4971, 5012, 5039, 5104, 5123, 5142, 5210, 5346-5354, 8482, 8612, 9027, 9030, 9030, 9044, 9048-9049, 9056, 9061, 9066, 9238, 9238, 248, 1248, 1312, 13121
21427	835, 927, 947, 1200, 1208, 4931, 9027, 13488, 17584
25473	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1252, 4386, 4909, 863, 4946, 4948, 4953, 4970-4971, 5012, 5123, 829029, 82,
25479	37, 273, 277-278, 280, 284-285, 297, 423, 437, 500, 813, 819, 836, 838, 850, 852, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 920, 1025-1027, 1041-1043, 1115, 1252, 4909, 4932, 4934, 4946, 4948, 4953, 4970-4971, 5012, 5123, 5211,
25480	37, 500, 904, 1114, 5210, 8229, 28709
25508	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25518, 29614, 33698-33700, 37796
25510	933-934, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29621, 33717, 37813
25512	935-936, 946, 1381, 5031, 5477, 5484, 9127, 13223
25514	937-938, 950, 1370, 5033, 5046, 9142
25518	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 29614, 33698-33700, 37796
25520	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25525, 29616, 29621, 33717, 37813
25524	937, 948, 950, 1370, 5033, 5046, 9142, 29620
25525	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 29616, 29621, 33717, 37813

Tabela 674. Obsługa konwersji identyfikatora CCSID produktu IBM MQ for z/OS (kontynuacja)	
CCSID	**Konwertuje do i z CCSID**
25617	37, 273, 277-278, 280, 284-285, 290, 297, 367, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 895-897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1252, 4386, 4909, 4951, 4932, 4934, 4946, 4948,
25619	37, 273, 277-278, 280, 284-285, 290, 297, 423, 437, 500, 813, 819, 833, 836, 838, 850, 852, 855, 857, 860-861, 863, 869-871, 874-875, 880, 897, 903, 912, 916, 1025-1027, 1040-1043, 1088, 1114, 4386, 4909, 49210, 4932, 4934, 4946, 4948, 4949, 4951, 53
25664	37, 273, 277-278, 280, 284-285, 290, 297, 367, 500, 819, 833, 836, 850, 852, 855, 857, 870-871, 875, 891, 1025-1027, 1040-1043, 1088, 1126, 4386, 4929, 4932, 4946, 4948, 4951, 4953, 4971, 5123, 8229, 8482, 9025, 9044, 9049, 13121, 25617, 25619, 28709
28709	37, 256, 273, 275, 277-278, 280, 284-285, 290, 297, 367, 420, 423-424, 437, 500, 720, 737, 775, 813, 819, 833, 836, 838, 850, 852, 855, 858, 860-866, 869-871, 874-875, 880, 897, 903-905, 912, 914-916, 920-924, 1009, 1025-1027, 11440-1043, 1047, 1047, 8612, 9025, 9030, 9044, 9049, 9056, 9061, 9066, 13121, 13488, 16804, 17248, 17584, 25473, 25479, 25480, 25617, 25619, 25664
29614	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 33698-33700, 37796
29616	933, 944, 949, 5029, 5045, 5460, 9125, 13221, 17317, 25520, 25525, 29621, 33717, 37813
29620	937, 948, 950, 1370, 5033, 5046, 9142, 25524
29621	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 33717, 37813
33698	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33699-33700, 37796
33699	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698, 33700, 37796
33700	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33699, 37796
33717	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 37813
37796	930-932, 939, 942-943, 1390, 1399, 5026, 5028, 5035, 5038-5039, 9122, 9124, 9131, 9135, 13218-13219, 13231, 17314, 25508, 25518, 29614, 33698-33700
37813	933-934, 944, 949, 1363-1364, 5029, 5045, 5460, 9125, 9555, 13221, 13651, 17317, 25510, 25520, 25525, 29616, 29621, 33717

IBM i Obsługa konwersji IBM i

Pełną listę identyfikatorów CCSID i konwersji obsługiwanych przez IBM imożna znaleźć w odpowiedniej publikacji IBM i .

Obsługiwane strony kodowe są wymienione w sekcji Obsługiwane odwzorowania CCSID.

Obsługa konwersji Unicode

Niektóre platformy obsługują konwersję danych użytkownika na kodowanie Unicode lub z kodowania Unicode. Obsługiwane są dwie formy kodowania Unicode: UTF-16 (CCSID 1200, 13488 i 17584) oraz UTF-8 (CCSID 1208). Należy użyć identyfikatorów CCSID 1200 lub 1208, ponieważ reprezentują one najnowszą obsługiwaną wersję kodu Unicode.

Obsługiwane są pary odpowiedników UTF-16 (para 2-bajtowych znaków UTF-16 w zakresie od X'D800'do X'DFFF', które reprezentują punkt kodowy Unicode powyżej U + FFFF). Jeśli docelowy identyfikator CCSID nie zawiera odwzorowania dla punktu kodowego reprezentowanego przez parę odpowiedników UTF-16 , para znaków jest przekształcana w pojedynczy znak podstawiany.

Sekwencje znaków łączących są obsługiwane przez IBM MQ. Oznacza to, że w niektórych przypadkach wstępnie złożony znak w źródłowym CCSID zostanie przekształcony w sekwencję znaków łączących w docelowym CCSID lub w inny sposób.

Uwaga: Produkt IBM MQ nie obsługuje identyfikatorów CCSID menedżera kolejek w formacie UTF-16 , dlatego dane nagłówka komunikatu nie mogą być kodowane w formacie UTF-16.

Obsługa kodu Unicode w produkcie IBM MQ AIX

AIX

W systemie IBM MQ for AIX konwersja do i z obsługiwanych identyfikatorów CCSID Unicode (najlepiej 1200 lub 1208) jest obsługiwana dla identyfikatorów CCSID innych niż Unicode na poniższej liście:

037
273, 278, 280, 284, 285, 297
423, 437
500
813, 819, 850, 852, 856, 857, 858, 860, 861, 865, 867, 869, 875, 878, 880
901, 902, 912, 915, 916, 920, 923, 924, 932, 933, 935, 937, 938, 939, 942, 943, 948, 949, 950, 954, 964, 970
1026, 1046, 1089
1129, 1130, 1131, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1156, 1157
1200, 1208, 1250, 1251, 1253, 1254, 1258, 1280, 1281, 1282, 1283, 1284, 1285
1363, 1364, 1381, 1383, 1386, 1388
4899
5026, 5035, 5050, 5346, 5347, 5348, 5349, 5350, 5351, 5352, 5353, 5354, 5488
9044, 9048, 9449
12712
13488
17584
33722

Obsługa kodu Unicode w systemach IBM MQ for Windows i Linux

Windows Linux

W systemach IBM MQ for Windows oraz IBM MQ dla konwersji do i z systemu Linux obsługiwane identyfikatory CCSID Unicode (najlepiej 1200 lub 1208) są obsługiwane dla identyfikatorów CCSID innych niż Unicode na poniższej liście:

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” na stronie 1018], 915, 916, 918, 920, 921, 922, 923, 924, 927, 928, 930, 931[“1” na stronie 1018], 932[“2” na stronie 1018], 933, 935, 937, 938[“3” na stronie 1018], 939, 941, 942, 943, 947, 948, 949, 950, 951, 954[“4” na stronie 1018], 964, 970
1006, 1025, 1026, 1027, 1040, 1041, 1042, 1043, 1046, 1047, 1051, 1088, 1089, 1097, 1098
1112, 1114, 1115, 1122, 1123, 1124, 1129, 1130, 1132, 1133, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1156, 1157
1200, 1208, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1275, 1280, 1281, 1282, 1283
1363, 1364, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1383, 1386, 1388
4899
5050, 5346, 5347, 5348, 5349, 5350, 5351, 5352, 5353, 5354, 5488[“5” na stronie 1018]
9044, 9048, 9449
12712
13488
17584
33722[“4” na stronie 1018]

Uwagi:

1. 931 używa 939 do konwersji.
2. 932 używa 942 do konwersji.
3. 938 używa 948 do konwersji.
4. Jednostki 954 i 33722 używają jednostki 5050 do konwersji.
5. Tylko w systemach Windows i Linux .

Obsługa kodu Unicode w systemie IBM i

IBM i

Szczegółowe informacje na temat obsługi kodu UNICODE można znaleźć w odpowiedniej publikacji IBM i dotyczącej danego systemu operacyjnego.

Obsługa kodu Unicode w systemie IBM MQ for z/OS

z/OS

W systemie IBM MQ for z/OS konwersja do i z obsługiwanych identyfikatorów CCSID Unicode (najlepiej 1200 lub 1208) jest obsługiwana dla identyfikatorów CCSID innych niż Unicode na poniższej liście:

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, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1025, 1026, 1027, 1040, 1041, 1042, 1043, 1046, 1047, 1051, 1088, 1089, 1097, 1098
1112, 1114, 1115, 1122, 1123, 1124, 1125, 1126, 1129, 1130, 1131, 1132, 1133, 1137, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1164
1200, 1208, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1275, 1276, 1277, 1280, 1281, 1282, 1283, 1284, 1285
1351, 1362, 1363, 1364, 1370, 1371, 1380, 1381, 1385, 1386, 1388, 1390, 1399
4899, 4909, 4930, 4933, 4948, 4951, 4952, 4960, 4971
5012 5039 5104 5123 5142 5210 5346 5347 5348 5349 5350 5351 5352 5353 5354 5488
8482 8612
9027 9030 9044 9048 9049 9056 9061 9066 9238 9449
1166
12712
13121, 13218, 13488, 1374, 1375, 1376, 1377, 1378, 1379
16684, 16804
17248, 17584
21427
28709

Standardy kodowania na platformach 64-bitowych

Ten temat zawiera informacje o standardach kodowania na platformach 64-bitowych i preferowanych typach danych.

Preferowane typy danych

Te typy nigdy nie zmieniają wielkości i są dostępne zarówno na 32-bitowych, jak i 64-bitowych platformach IBM MQ :

Tabela 675. Nazwy i długości typów danych

Nazwa	Długość
MQLONG	4 bajty
MQULONG	4 bajty
MQINT32	4 bajty
MQUINT32	4 bajty
MQINT64	8 bajtów
MQUINT64	8 bajtów

ALW Standardowe typy danych w systemie AIX, Linux, and Windows

Informacje o standardowych typach danych w 32-bitowych aplikacjach AIX and Linuxi 64-bitowych aplikacjach AIX, Linux, and Windows .

32-bitowe aplikacje AIX and Linux

Linux AIX

Tabela 676. Nazwy i długości typów danych dla 32-bitowych aplikacji AIX and Linux

Nazwa	Długość
char	1 bajt
short	2 bajty
int	4 bajty
long	4 bajty
liczba zmiennopozycyjna	4 bajty
double (podwójna)	8 bajtów
long double	8 bajtów
problemów dotyczących	4 bajty
ptrdiff_ t	4 bajty
wielkość_t	4 bajty
czas t	4 bajty
zegar_t	4 bajty
wchar_t	4 bajty

AIX Należy zauważyć, że w systemie AIX wartość wchar_t wynosi 2 bajty.

64-bitowe aplikacje AIX and Linux

Linux AIX

Tabela 677. Nazwy i długości typów danych dla 64-bitowych aplikacji AIX and Linux

Nazwa	Długość
char	1 bajt
short	2 bajty
int	4 bajty
long	8 bajtów
liczba zmiennopozycyjna	4 bajty
double (podwójna)	8 bajtów
long double	8 bajtów
problemów dotyczących	8 bajtów
ptrdiff_ t	8 bajtów
wielkość_t	8 bajtów
czas t	8 bajtów
zegar_t	4 bajty
wchar_t	4 bajty

AIX Należy zauważyć, że w systemie AIX wartość wchar_t wynosi 2 bajty.

Windows (aplikacje 64-bitowe)

Windows

Tabela 678. Nazwy i długości typów danych dla 64-bitowych aplikacji Windows

Nazwa	Długość
char	1 bajt
short	2 bajty
int	4 bajty
long	4 bajty
liczba zmiennopozycyjna	4 bajty
double (podwójna)	8 bajtów
long double	8 bajtów
problemów dotyczących	8 bajtów
	Należy zauważyć, że wszystkie wskaźniki mają długość 8 bajtów.
ptrdiff_ t	8 bajtów
wielkość_t	8 bajtów
czas t	8 bajtów
zegar_t	4 bajty
wchar_t	2 bajty
Word	2 bajty
DWORD	4 bajty
aplikacji	8 bajtów
HPLIK	4 bajty

Uwagi dotyczące kodowania w systemie Windows

Windows

HANDLE hf;

Użycie

```
hf = CreateFile((LPCTSTR) FileName,
                Access,
                ShareMode,
                xihSecAttsNTRestrict,
                Create,
                AttrAndFlags,
                NULL);
```

Nie używaj

```
HFILE hf;
hf = (HFILE) CreateFile((LPCTSTR) FileName,
                        Access,
                        ShareMode,
                        xihSecAttsNTRestrict,
                        Create,
                        AttrAndFlags,
                        NULL);
```

ponieważ powoduje to błąd.

rozmiar_t len fgets

Użycie

```
size_t len
while (fgets(string1, (int) len, fp) != NULL)
len = strlen(buffer);
```

Nie używaj

```
int len;

while (fgets(string1, len, fp) != NULL)
len = strlen(buffer);
```

printf,

Użycie

```
printf("My struc pointer: %p", pMyStruc);
```

Nie używaj

```
printf("My struc pointer: %x", pMyStruc);
```

Jeśli potrzebne są dane szesnastkowe, należy drukować 4 bajty górne i dolne oddzielnie.

znak * ptr

Użycie

```
char * ptr1;
char * ptr2;
size_t bufLen;

bufLen = ptr2 - ptr1;
```

Nie używaj

```
char *ptr1;
char *ptr2;
UINT32 bufLen;

bufLen = ptr2 - ptr1;
```

alignBytes

Użycie

```
alignBytes = (unsigned short) ((size_t) address % 16);
```

Nie używaj

```
void *address;
unsigned short alignBytes;

alignBytes = (unsigned short) ((UINT32) address % 16);
```

len

Użycie

```
len = (UINT32) ((char *) address2 - (char *) address1);
```

Nie używaj

```
void *address1;
void *address2;
UINT32 len;

len = (UINT32) ((char *) address2 - (char *) address1);
```

sscanf,

Użycie

```
MQLONG SBCSprt;

sscanf(line, "%d", &SBCSprt);
```

Nie używaj

```
MQLONG SBCSprt;

sscanf(line, "%1d", &SBCSprt);
```

Program `%ld` próbuje umieścić typ 8-bajtowy w typie 4-bajtowym. Należy używać parametru `%l` tylko wtedy, gdy używany jest rzeczywisty typ danych `long` . MQLONG, UINT32 i INT32 są zdefiniowane jako cztery bajty, tak samo jak `int` na wszystkich platformach IBM MQ :

IBM i IBM i Skorowidz programistyczny aplikacji (ILE/RPG)

Programowanie aplikacji dla systemu IBM i.

Te informacje są pomocne podczas tworzenia aplikacji dla produktu IBM i.

- “Opisy typów danych w systemie IBM i” na stronie 1024
- “Wywołania funkcji w systemie IBM i” na stronie 1289
- “Atrybuty obiektów w systemie IBM i” na stronie 1410
- “Aplikacje” na stronie 1458
- “Kody powrotu dla IBM i (ILE RPG)” na stronie 1471
- “Reguły sprawdzania poprawności opcji MQI dla IBM i (ILE RPG)” na stronie 1473
- “Kodowania maszynowe w systemie IBM i” na stronie 1475
- “Opcje raportu i flagi komunikatów w systemie IBM i” na stronie 1478

Dezaktualizacja trybu zgodności dla aplikacji RPG i COBOL w systemie IBM i

IBM i

Począwszy od wersji IBM MQ for IBM i 9.0, produkt nie udostępnia już obsługi dla aplikacji RPG lub COBOL, które korzystają z łączenia dynamicznego nazywanego trybem zgodności. Ten tryb działania był wymagany w przypadku aplikacji napisanych przed wersją MQSeries 5.1, a kolejne wersje produktu udostępniały kompatybilne środowisko wykonawcze dla tych aplikacji, nawet jeśli struktury copybook potrzebne do ich skompilowania zostały usunięte w produkcie IBM WebSphere MQ 6.0. Połączenie dynamiczne (tryb zgodności) zostało udostępnione przez następujące programy w bibliotece QMQM, które zostały usunięte w katalogu IBM MQ for IBM i 9.0:

- Komenda AMQVSTUB
- Komunikat AMQZSTUB
- QMQM,
- MQCLOSE
- ZMQCONN
- MQDISC

- MQGET
- MQINQ
- MQOPEN
- MQPUT
- MQPUT1
- MQSET

W produkcie IBM MQ for IBM i 9.0aplikacje korzystające z tego trybu zgodności muszą zostać zrekompilowane, aby można było używać wywołań MQ powiązanych statycznie, które są udostępniane przez programy usługowe LIBMQM i LIBMQM_R. Przykładowe programy, takie jak AMQ3PUT4 i AMQ3GET4, przedstawiają sposób użycia tego modelu programistycznego. Więcej informacji na temat używania tych wywołań produktu MQ zawiera podręcznik IBM i Application Programming Reference (ILE/RPG)(Skorowidz programistyczny aplikacji w systemie IBM i-ILE/RPG).

Uwagi:

- Aby użyć programu usługowego LIBMQM, należy przekodować aplikacje, które obecnie używają interfejsu CALL 'QMQM'.

  Obiekty programów i programy usługowe z poprzedniej listy, na przykład QMQM, MQCONN, MQPUT, AMQVSTUB i AMQZSTUB, są usuwane w produkcie IBM MQ for IBM i 9.0, a aplikacje, które zostały zakodowane do używania trybu zgodności, przestają działać.
- Jeśli aplikacje są powiązane z programem usługowym LIBMQM pod adresem IBM MQ for IBM i 8.0, nie trzeba rekompilować ani ponownie konsolidować tych aplikacji w wersji IBM MQ for IBM i 9.0 lub nowszej.
- Na tej samej partycji nie można zainstalować więcej niż jednej wersji systemu IBM MQ for IBM i .

Aby dowiedzieć się, czy program w języku RPG lub COBOL używa trybu zgodności, należy użyć komendy **DSPPGMREF** (Display Program References-Wyświetlenie odniesień do programów) w celu wyświetlenia programów zewnętrznych wywoływanych przez program użytkowy. Jeśli istnieją odwołania do programów wymienionych w tej sekcji, program nie zostanie uruchomiony w wersji IBM MQ for IBM i 9.0 lub nowszej. W poniższym przykładzie danych wyjściowych komendy **DSPPGMREF** przedstawiono trzy nieaktualne obiekty programu MQCONN, MQOPEN i MQCLOSE:

```
Program . . . . . :  MYAPPPGM
Library . . . . . :    MYLIB
Text 'description'. . . . . :     ILE/COBOL SAMPLE PUT TO QUEUE (MQPUT)
Number of objects referenced  . . . . . :       5
Object  . . . . . :    MQCONN
Library . . . . . :      *LIBL
Object type . . . . . :        *PGM
Object  . . . . . :    MQOPEN
Library . . . . . :      *LIBL
Object type . . . . . :        *PGM
Object  . . . . . :    MQCLOSE
Library . . . . . :      *LIBL
Object type . . . . . :        *PGM
```

Takie programy muszą zostać zrekompilowane przy użyciu metody Bound Proceduralne Call opisanej w sekcji Przygotowywanie programów w języku COBOL w produkcie IBM i.

W przypadku próby uruchomienia aplikacji w systemie IBM MQ for IBM i 9.0 lub nowszym korzystającej z trybu zgodności, najczęściej występującym pierwszym błędem jest błąd MCH3401 podczas próby wywołania programu MQCONN lub QMQM.

Zadania pokrewne

Projektowanie aplikacji

IBM i Opisy typów danych w systemie IBM i

Ta kolekcja tematów zawiera opisy typów danych używanych w programowaniu w języku IBM i .

Konwencje stosowane w opisie typów danych

Dla każdego podstawowego typu danych informacje te zawierają opis jego użycia, w formie niezależnej od języka programowania. Po tym następują typowe deklaracje w wersji ILE języka programowania RPG. W tym miejscu przedstawiono definicje podstawowych typów danych w celu zapewnienia spójności. Język RPG używa specyfikacji 'D', w których pola robocze mogą być deklarowane przy użyciu dowolnych potrzebnych atrybutów. Można to jednak zrobić w specyfikacjach obliczeniowych, w których pole jest używane.

Aby użyć podstawowych typów danych, należy utworzyć:

- Element /COPY zawierający wszystkie typy danych, lub
- Zewnętrzna struktura danych (PF) zawierająca wszystkie typy danych. Następnie należy określić pola robocze z atrybutami LIKE w odpowiednim polu typu danych.

Korzyścią z drugiej opcji jest to, że definicje mogą być używane jako 'FIELD REFERENCE FILE' dla innych obiektów IBM i . Jeśli definicja typu danych IBM MQ ulegnie zmianie, ponowne utworzenie tych obiektów jest stosunkowo proste.

Podstawowe typy danych

Wszystkie inne typy danych opisane w tej sekcji odpowiadają bezpośrednio tym elementarnym typom danych lub zagregowanym elementarnym typom danych (tablicom lub strukturom).

Tabela 679. Podstawowe typy danych

Typ danych	**Reprezentacja**
MQBOOL	10-cyfrowa liczba całkowita ze znakiem
MQBYTE	1-bajtowe pole alfanumeryczne
MQBYTE16	16-bajtowe pole alfanumeryczne
MQBYTE24	24-bajtowe pole alfanumeryczne
MQBYTE32	32-bajtowe pole alfanumeryczne
MQBYTE64	64-bajtowe pole alfanumeryczne
MQCHAR	1-bajtowe pole alfanumeryczne
MQCHAR4	4-bajtowe pole alfanumeryczne
MQCHAR8	8-bajtowe pole alfanumeryczne
MQCHAR12	12-bajtowe pole alfanumeryczne
MQCHAR16	16-bajtowe pole alfanumeryczne
MQCHAR20	20-bajtowe pole alfanumeryczne
MQCHAR28	28-bajtowe pole alfanumeryczne
MQCHAR32	32-bajtowe pole alfanumeryczne
MQCHAR48	48-bajtowe pole alfanumeryczne
MQCHAR64	64-bajtowe pole alfanumeryczne
MQCHAR128	128-bajtowe pole alfanumeryczne
MQCHAR256	256-bajtowe pole alfanumeryczne
MQFLOAT32	4-bajtowa liczba zmiennopozycyjna
MQFLOAT64	8-bajtowa liczba zmiennopozycyjna

Tabela 679. Podstawowe typy danych (kontynuacja)	
Typ danych	**Reprezentacja**
MQHCONFIG (konfiguracja MQ)	Uchwyt konfiguracji
MQHCONN	10-cyfrowa liczba całkowita ze znakiem
MQHMSG	Uchwyt komunikatu nadający dostęp do komunikatu
MQHOBJ	10-cyfrowa liczba całkowita ze znakiem
MQINT8	8-bitowa liczba całkowita ze znakiem
MQINT16	16-bitowa liczba całkowita ze znakiem
MQINT32	32-bitowa liczba całkowita ze znakiem
MQINT64	64-bitowa liczba całkowita ze znakiem
MQLONG	32-bitowa liczba całkowita ze znakiem
Identyfikator MQPID	Identyfikator procesu
MQPTR	Pointer
ID_zmaterializowanej tabeli zapytania	Identyfikator wątku
MQUINT8	8-bitowa liczba całkowita bez znaku
MQUINT16	16-bitowa liczba całkowita bez znaku
MQUINT32	32-bitowa liczba całkowita bez znaku
MQUINT64	64-bitowa liczba całkowita bez znaku
MQULONG	32-bitowa liczba całkowita bez znaku
PMQACH	Wskaźnik do struktury danych typu MQACH
PMQAIR	Wskaźnik do struktury danych typu MQAIR
PMQAXC	Wskaźnik do struktury danych typu MQAXC
PMAXP	Wskaźnik do struktury danych typu MAXP
PMQBMHO	Wskaźnik do struktury danych typu MQBMHO
PMQBO	Wskaźnik do struktury danych typu MQBO
PMQBOOL	Wskaźnik do danych typu MQBOOL
PMQBYTE,	Wskaźnik do danych typu MQBYTE
PMQBYTEN	Wskaźnik do danych typu MQBYTEn
PMQCBC,	Wskaźnik do struktury danych typu MQCBC
PMQCBD,	Wskaźnik do struktury danych typu MQCBD
PMQCHAR	Wskaźnik do struktury danych typu MQCHAR
PMQCHARV	Wskaźnik do struktury danych typu MQCHARV
PMQCHARn	Wskaźnik do danych typu MQCHARn
PMQCIH	Wskaźnik do struktury danych typu MQCIH
PMQCMHO	Wskaźnik do struktury danych typu MQCMHO
PMQCNO	Wskaźnik do struktury danych typu MQCNO

Tabela 679. Podstawowe typy danych (kontynuacja)	
Typ danych	**Reprezentacja**
Protokół PMQCSP	Wskaźnik do struktury danych typu MQCSP
PMQCTLO	Wskaźnik do struktury danych typu MQCTLO
PMQDH	Wskaźnik do struktury danych typu MQDH
PMQDHO	Wskaźnik do struktury danych typu MQDHO
PMQDLH	Wskaźnik do struktury danych typu MQDLH
PMQDMHO	Wskaźnik do struktury danych typu MQDMHO
PMQDMPO	Wskaźnik do struktury danych typu MQDMPO
PMQEPH	Wskaźnik do struktury danych typu MQEPH
PMQFLOAT32	Wskaźnik do danych typu MQFLOAT32
PMQFLOAT64	Wskaźnik do danych typu MQFLOAT64
PMQFUNC	Wskaźnik do funkcji
PMQGMO	Wskaźnik do struktury danych typu MQGMO
Komenda PMQHCONFIG	Wskaźnik do danych typu MQHCONFIG
PMQHCONN	Wskaźnik do danych typu MQHCONN
PMQHMSG	Wskaźnik do danych typu MQHMSG
PMQHOBJ	Wskaźnik do danych typu MQHOBJ
PMQIIH	Wskaźnik do struktury danych typu MQIIH
PMQIMPO	Wskaźnik do struktury danych typu MQIMPO
PMQINT8	Wskaźnik do danych typu MQINT8
PMQINT16	Wskaźnik do danych typu MQINT16
PMQINT32	Wskaźnik do danych typu MQINT32
PMQINT64	Wskaźnik do danych typu MQINT64
PMQLONG	Wskaźnik do danych typu MQLONG
PMQMD	Wskaźnik do struktury danych typu MQMD
PMQMDE,	Wskaźnik do struktury danych typu MQMDE
PMQMD1	Wskaźnik do struktury danych typu MQMD1
PMQMD2	Wskaźnik do struktury danych typu MQMD2
PMQMHBO,	Wskaźnik do struktury danych typu MQMHBO
PMQOD	Wskaźnik do struktury danych typu MQOD
PMQOR,	Wskaźnik do struktury danych typu MQOR
Komenda PMQPD	Wskaźnik do struktury danych typu MQPD
PMQPID	Wskaźnik do identyfikatora procesu MQPID
PMQPMO	Wskaźnik do struktury danych typu MQPMO
PMQPTR	Wskaźnik do danych typu MQPTR

Tabela 679. Podstawowe typy danych (kontynuacja)	
Typ danych	**Reprezentacja**
PMQRFH	Wskaźnik do struktury danych typu MQRFH
PMQRFH2	Wskaźnik do struktury danych typu MQRFH2
PMQRMH	Wskaźnik do struktury danych typu MQRMH
PMQRR	Wskaźnik do struktury danych typu MQRR
PMQSCO	Wskaźnik do struktury danych typu MQSCO
PMQSD	Wskaźnik do struktury danych typu MQSD
PMQSMPO	Wskaźnik do struktury danych typu MQSMPO
PMQSRO	Wskaźnik do struktury danych typu MQSRO
PMQSTS	Wskaźnik do struktury danych typu MQSTS
Identyfikator PMQTID	Wskaźnik do identyfikatora wątku MQTID
PMQTM,	Wskaźnik do struktury danych typu MQTM
PMQTMC2	Wskaźnik do struktury danych typu MQTMC2
PMQUINT8	Wskaźnik do danych typu MQUINT8
PMQUINT16	Wskaźnik do danych typu MQUINT16
PMQUINT32	Wskaźnik do danych typu MQUINT32
PMQUINT64	Wskaźnik do danych typu MQUINT64
PMQULONG	Wskaźnik do danych typu MQULONG
PMQVOID (identyfikator produktu MQ)	Pointer
PMQWIH	Wskaźnik do struktury danych typu MQWIH
PMQXQH	Wskaźnik do struktury danych typu MQXQH

IBM i *MQBOOL w systemie IBM i*

Typ danych MQBOOL reprezentuje wartość boolowską. Wartość 0 oznacza fałsz. Każda inna wartość reprezentuje wartość true.

Obiekt MQBOOL musi być wyrównany tak, jak w przypadku typu danych MQLONG.

IBM i *MQBYTE w systemie IBM i*

Typ danych MQBYTE reprezentuje jeden bajt danych.

Na bajcie nie jest umieszczana żadna konkretna interpretacja-jest ona traktowana jako łańcuch bitów, a nie jako liczba lub znak binarny. Nie jest wymagane żadne specjalne wyrównanie.

Tablica danych MQBYTE jest czasami używana do reprezentowania obszaru pamięci głównej o charakterze, który nie jest znany menedżerowi kolejek. Na przykład obszar może zawierać dane komunikatu aplikacji lub strukturę. Wyrównanie granic tego obszaru musi być zgodne z charakterem zawartych w nim danych.

IBM i *MQBYTEn (łańcuch n bajtów) w systemie IBM i*

Każdy typ danych MQBYTEn reprezentuje łańcuch *n* bajtów.

Gdzie *n* może przyjmować jedną z następujących wartości:

- 16, 24, 32 lub 64.

Każdy bajt jest opisany przez typ danych MQBYTE. Nie jest wymagane żadne specjalne wyrównanie.

Jeśli dane w łańcuchu są krótsze niż zdefiniowana długość łańcucha, dane muszą być dopełnione wartościami pustymi, aby można było wypełnić łańcuch.

Gdy menedżer kolejek zwraca do aplikacji łańcuchy bajtów (na przykład w wywołaniu MQGET), menedżer kolejek zawsze jest dopełniany wartościami pustymi do zdefiniowanej długości łańcucha.

Dostępne są stałe definiujące długości pól łańcuchów bajtowych.

IBM i *MQCHAR (znak) w systemie IBM i*

Typ danych MQCHAR reprezentuje pojedynczy znak.

Identyfikator kodowanego zestawu znaków jest taki, jak identyfikator menedżera kolejek (patrz atrybut **CodedCharSetId** w temacie CodedCharSetId). Nie jest wymagane żadne specjalne wyrównanie.

Uwaga: Dane komunikatu aplikacji określone w wywołaniach MQGET, MQPUT i MQPUT1 są opisane przez typ danych MQBYTE, a nie przez typ danych MQCHAR.

IBM i *MQCHARn (łańcuch n znaków) w systemie IBM i*

Każdy typ danych MQCHARn reprezentuje łańcuch *n* znaków.

Gdzie *n* może przyjmować jedną z następujących wartości:

- 4, 8, 12, 16, 20, 28, 32, 48, 64, 128 lub 256

Każdy znak jest opisywany przez typ danych MQCHAR. Nie jest wymagane żadne specjalne wyrównanie.

Jeśli dane w łańcuchu są krótsze niż zdefiniowana długość łańcucha, dane muszą być uzupełnione odstępami, aby można było wypełnić łańcuch. W niektórych przypadkach do przedwczesnego zakończenia łańcucha zamiast dopełniania odstępami można użyć znaku o kodzie zero; znak o kodzie zero i następujące po nim znaki są traktowane jako odstępy aż do zdefiniowanej długości łańcucha. Miejsca, w których można używać wartości NULL, są identyfikowane w opisach wywołania i typu danych.

Gdy menedżer kolejek zwraca łańcuchy znaków do aplikacji (na przykład w wywołaniu MQGET), menedżer kolejek zawsze dopełnia znakami spacji do zdefiniowanej długości łańcucha. Menedżer kolejek nie używa znaku o kodzie zero do rozgraniczenia łańcucha.

Dostępne są stałe definiujące długości pól łańcucha znaków.

IBM i *MQFLOAT32 w systemie IBM i*

Typ danych MQFLOAT32 to 32-bitowa liczba zmiennopozycyjna reprezentowana przy użyciu standardowego formatu zmiennopozycyjnego IEEE.

Obiekt MQFLOAT32 musi być wyrównany do granicy 4-bajtowej.

IBM i *MQFLOAT64 w systemie IBM i*

Typ danych MQFLOAT64 to 64-bitowa liczba zmiennopozycyjna reprezentowana przy użyciu standardowego formatu zmiennopozycyjnego IEEE.

Wartość MQFLOAT64 musi być wyrównana do granicy 8-bajtowej.

MQHCONFIG-uchwyt konfiguracji

Typ danych MQHCONFIG reprezentuje uchwyt konfiguracji, czyli komponent, który jest konfigurowany dla konkretnej instalowalnej usługi. Uchwyt konfiguracji musi być wyrównany do swojej granicy naturalnej.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko pod kątem równości.

IBM i *MQHCONN (uchwyt połączenia) w systemie IBM i*

Typ danych MQHCONN reprezentuje uchwyt połączenia, czyli połączenie z konkretnym menedżerem kolejek.

Uchwyt połączenia musi być wyrównany do jego naturalnej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko pod kątem równości.

IBM i *MQHMSG (uchwyt komunikatu) w systemie IBM i*

Typ danych **MQHMSG** reprezentuje uchwyt komunikatu, który zapewnia dostęp do komunikatu.

Uchwyt komunikatu musi być wyrównany do 8-bajtowej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko pod kątem równości.

IBM i *MQHOBJ (uchwyt obiektu) w systemie IBM i*

Typ danych MQHOBJ reprezentuje uchwyt obiektu, który daje dostęp do obiektu.

Uchwyt obiektu musi być wyrównany do jego naturalnej granicy.

Uwaga: Aplikacje muszą testować zmienne tego typu tylko pod kątem równości.

IBM i *MQINT8 (8-bitowa liczba całkowita ze znakiem) w systemie IBM i*

Typ danych MQINT8 to 8-bitowa liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -128 do +127, chyba że kontekst ogranicza inaczej.

IBM i *MQINT16 (16-bitowa liczba całkowita ze znakiem) w systemie IBM i*

Typ danych MQINT16 to 16-bitowa liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -32 768 do +32 767, chyba że kontekst stanowi inaczej.

Obiekt MQINT16 musi być wyrównany do granicy dwubajtowej.

IBM i *MQINT32 (32-bitowa liczba całkowita) w systemie IBM i*

Typ danych MQINT32 to 32-bitowa liczba całkowita ze znakiem.

Jest to odpowiednik MQLONG.

IBM i *MQINT64 (64-bitowa liczba całkowita) w systemie IBM i*

Typ danych MQINT64 to 64-bitowa liczba całkowita ze znakiem, która może przyjmować dowolną wartość z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, o ile nie jest to ograniczone przez kontekst.

Dla języka COBOL poprawny zakres jest ograniczony do -999 999 999 999 999 999 do +999 999 999 999 999 999. Wartość MQINT64 powinna być wyrównana do granicy 8-bajtowej.

IBM i *MQLONG (długa liczba całkowita) w systemie IBM i*

Typ danych MQLONG jest 32-bitową binarną liczbą całkowitą ze znakiem, która może przyjmować dowolną wartość z zakresu od -2 147 483 648 do + 2 147 483 647, o ile nie jest to ograniczone przez kontekst, wyrównaną do jego granicy naturalnej.

MQPID-identyfikator procesu

Identyfikator procesu IBM MQ .

Jest to ten sam identyfikator, który jest używany w śledzeniu IBM MQ i zrzutach FFST , ale może być inny niż identyfikator procesu systemu operacyjnego.

MQPTR-wskaźnik

Typ danych MQPTR to adres danych dowolnego typu. Wskaźnik musi być wyrównany względem swojej granicy naturalnej; jest to granica 16-bajtowa w systemie IBM i.

Niektóre języki programowania obsługują wskaźniki określonego typu. MQI również używa tych wskaźników w kilku przypadkach.

MQTID-identyfikator wątku
Identyfikator wątku MQ .

Jest to ten sam identyfikator, który jest używany w zrzutach MQ śledzenia i FFST , ale może być inny niż identyfikator wątku systemu operacyjnego.

IBM i *MQUINT8 (8-bitowa liczba całkowita bez znaku) w systemie IBM i*
Typ danych MQUINT8 to 8-bitowa liczba całkowita bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +255, o ile nie jest to ograniczone przez kontekst.

MQUINT16 -16-bitowa liczba całkowita bez znaku
Typ danych MQUINT16 to 16-bitowa liczba całkowita bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +65 535, chyba że kontekst ogranicza inaczej.

Wartość MQUINT16 musi być wyrównana do granicy dwubajtowej.

IBM i *MQUINT32 (32-bitowa liczba całkowita bez znaku) w systemie IBM i*
Typ danych MQUINT32 to 32-bitowa liczba całkowita bez znaku. Jest to odpowiednik komendy MQULONG.

MQUINT64 -64-bitowa liczba całkowita bez znaku
Typ danych MQUINT64 jest 64-bitową liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do +18 446 744 073 709 551 615, chyba że kontekst stanowi inaczej.

Dla języka COBOL poprawny zakres jest ograniczony do zakresu od 0 do +999 999 999 999 999. Wartość MQUINT64 powinna być wyrównana do granicy 8-bajtowej.

MQULONG-32-bitowa liczba całkowita bez znaku
Typ danych MQULONG jest 32-bitową binarną liczbą całkowitą bez znaku, która może przyjmować dowolną wartość z zakresu od 0 do + 4 294 967 294, chyba że kontekst ogranicza inaczej.

Obiekt MQULONG musi być wyrównany do granicy 4-bajtowej.

PMQACH-wskaźnik do struktury danych typu MQACH
Wskaźnik do struktury danych typu MQACH.

PMQAIR-wskaźnik do struktury danych typu MQAIR
Wskaźnik do struktury danych typu MQAIR.

PMQAXC-wskaźnik do struktury danych typu MQAXC
Wskaźnik do struktury danych typu MQAXC.

PMQAXP-wskaźnik do struktury danych typu MQAXP
Wskaźnik do struktury danych typu MQAXP.

PMQBMHO-wskaźnik do struktury danych typu MQBMHO
Wskaźnik do struktury danych typu MQBMHO.

PMQBO-wskaźnik do struktury danych typu MQBO
Wskaźnik do struktury danych typu MQBO.

PMQBOOL-wskaźnik do danych typu MQBOOL
Wskaźnik do danych typu MQBOOL.

Wskaźnik do danych typu MQBOOL.

PMQBYTE-wskaźnik do typu danych MQBYTE
Wskaźnik do typu danych MQBYTE.

PMQBYTEn-wskaźnik do struktury danych typu MQBYTEn
Wskaźnik do struktury danych typu MQBYTEn, gdzie n może mieć wartość 8, 12, 16, 24, 32, 40, 48 lub 128.

PMQCBC-wskaźnik do struktury danych typu MQCBC
Wskaźnik do struktury danych typu MQCBC.

PMQCBD-wskaźnik do struktury danych typu MQCBD
Wskaźnik do struktury danych typu MQCBD.

PMQCHAR-wskaźnik do danych typu MQCHAR
Wskaźnik do danych typu MQCHAR.

PMQCHARV-wskaźnik do struktury danych typu MQCHARV
Wskaźnik do struktury danych typu MQCHARV.

PMQCHARn-wskaźnik do typu danych MQCHARn
Wskaźnik do typu danych MQCHARn, gdzie n może mieć wartość 4, 8, 12, 20, 28, 32, 64, 128, 256, 264.

PMQCIH-wskaźnik do struktury danych typu MQCIH
Wskaźnik do struktury danych typu MQCIH.

PMQCMHO-wskaźnik do struktury danych typu MQCMHO
Wskaźnik do struktury danych typu MQCMHO.

PMQCNO-wskaźnik do struktury danych typu MQCNO
Wskaźnik do struktury danych typu MQCNO.

PMQCSP-wskaźnik do struktury danych typu MQCSP
Wskaźnik do struktury danych typu MQCSP.

PMQCTLO-wskaźnik do struktury danych typu MQCTLO
Wskaźnik do struktury danych typu MQCTLO.

PMQDH-wskaźnik do struktury danych typu MQDH
Wskaźnik do struktury danych typu MQDH.

PMQDHO-wskaźnik do struktury danych typu MQDHO
Wskaźnik do struktury danych typu MQDHO.

PMQDLH-wskaźnik do struktury danych typu MQDLH
Wskaźnik do struktury danych typu MQDLH.

PMQDMHO-wskaźnik do struktury danych typu MQDMHO
Wskaźnik do struktury danych typu MQDMHO.

PMQDMPO-wskaźnik do struktury danych typu MQDMPO
Wskaźnik do struktury danych typu MQDMPO.

Wskaźnik do struktury danych typu MQDMPO.

PMQEPH-wskaźnik do struktury danych typu MQEPH
Wskaźnik do struktury danych typu MQEPH.

PMQFLOAT32 -wskaźnik do danych typu MQFLOAT32
Wskaźnik do danych typu MQFLOAT32.

PMQFLOAT64 -wskaźnik do danych typu MQFLOAT64
Wskaźnik do danych typu MQFLOAT64.

PMQFUNC-wskaźnik do funkcji
Wskaźnik do funkcji.

PMQGMO-wskaźnik do struktury danych typu MQGMO
Wskaźnik do struktury danych typu MQGMO.

PMQHCONFIG-wskaźnik do typu danych MQHCONFIG
Wskaźnik do typu danych MQHCONFIG.

PMQHCONN-wskaźnik do typu danych MQHCONN
Wskaźnik do typu danych MQHCONN.

PMQHMSG-wskaźnik do typu danych MQHMSG
Wskaźnik do typu danych MQHMSG.

PMQHOBJ-wskaźnik do danych typu MQHOBJ
Wskaźnik do danych typu MQSMPO.

PMQIIH-wskaźnik do struktury danych typu MQIIH
Wskaźnik do struktury danych typu MQIIH.

PMQIMPO-wskaźnik do struktury danych typu MQIMPO
Wskaźnik do struktury danych typu MQIMPO.

PMQINT8 -wskaźnik do danych typu MQINT8
Wskaźnik do danych typu MQINT8.

PMQINT16 -wskaźnik do danych typu MQINT16
Wskaźnik do danych typu MQINT16.

IBM i *PMQINT32 (wskaźnik do danych typu MQINT32) w systemie IBM i*
Typ danych PMQINT32 jest wskaźnikiem do danych typu MQINT32. Jest to odpowiednik PMQLONG.

IBM i *PMQINT64 (wskaźnik do danych typu MQINT64) w systemie IBM i*
Typ danych PMQINT64 jest wskaźnikiem do danych typu MQINT64.

PMQLONG-wskaźnik do danych typu MQLONG
Wskaźnik do danych typu MQLONG.

PMQMD-wskaźnik do struktury typu MQMD
Wskaźnik do struktury typu MQMD.

PMQMDE-wskaźnik do struktury danych typu MQMDE
Wskaźnik do struktury danych typu MQMDE.

PMQMDI-wskaźnik do struktury danych typu MQMDI
Wskaźnik do struktury danych typu MQMDI.

PMQMD2 -wskaźnik do struktury danych typu MQMD2
Wskaźnik do struktury danych typu MQMD2

PMQMHBO-wskaźnik do struktury danych typu MQMHBO
Wskaźnik do struktury danych typu MQMHBO.

PMQOD-wskaźnik do struktury danych typu MQOD
Wskaźnik do struktury danych typu MQOD.

PMQOR-wskaźnik do struktury danych typu MQOR
Wskaźnik do struktury danych typu MQOR.

PMQPD-wskaźnik do struktury danych typu MQPD
Wskaźnik do struktury danych typu MQPD.

PMQPID-wskaźnik do identyfikatora procesu
Wskaźnik do identyfikatora procesu.

PMQPMO-wskaźnik do struktury danych typu MQPMO
Wskaźnik do struktury danych typu MQPMO.

PMQPTR-wskaźnik do danych typu MQPTR
Wskaźnik do danych typu MQPTR.

PMQRFH-wskaźnik do struktury danych typu MQRFH
Wskaźnik do struktury danych typu MQRFH.

PMQRFH2 -wskaźnik do struktury danych typu MQRFH2
Wskaźnik do struktury danych typu MQRFH2.

.

PMQRMH-wskaźnik do struktury danych typu MQRMH
Wskaźnik do struktury danych typu MQRMH.

PMQRR-wskaźnik do struktury danych typu MQRR
Wskaźnik do struktury danych typu MQRR.

PMQSCO-wskaźnik do struktury danych typu MQSCO
Wskaźnik do struktury danych typu MQSCO.

.

PMQSD-wskaźnik do struktury danych typu MQSD
Wskaźnik do struktury danych typu MQSD.

PMQSMPO-wskaźnik do struktury danych typu MQSMPO
Wskaźnik do struktury danych typu MQSMPO.

PMQSRO-wskaźnik do struktury danych typu MQSRO
Wskaźnik do struktury danych typu MQSRO.

PMQSTS-wskaźnik do struktury danych typu MQSTS
Wskaźnik do struktury danych typu MQSTS.

PMQTID-wskaźnik do struktury danych typu MQTID
Wskaźnik do struktury danych typu MQTID.

PMQTM-wskaźnik do struktury danych typu MQTM
Wskaźnik do struktury danych typu MQTM.

PMQTMC2 -wskaźnik do struktury danych typu MQTMC2
Wskaźnik do struktury danych typu MQTMC2.

PMQUINT8 -wskaźnik do danych typu MQUINT8
Wskaźnik do danych typu MQUINT8.

PMQUINT16 -wskaźnik do danych typu MQUINT16
Wskaźnik do danych typu MQUINT16.

IBM i *PMQUINT32 (wskaźnik do danych typu MQUINT32) w systemie IBM i*
Typ danych PMQUINT32 jest wskaźnikiem do danych typu MQUINT32. Jest to odpowiednik PMQULONG.

IBM i *PMQUINT64 (wskaźnik do danych typu MQUINT64) w systemie IBM i*
Typ danych PMQUINT64 jest wskaźnikiem do danych typu MQUINT64.

PMQULONG-wskaźnik do danych typu MQULONG
Wskaźnik do danych typu MQULONG.

PMQVOID-wskaźnik
Wskaźnik.

PMQWIH-wskaźnik do struktury danych typu MQWIH
Wskaźnik do struktury danych typu MQWIH.

PMQXQH-wskaźnik do struktury danych typu MQXQH
Wskaźnik do struktury danych typu MQXQH.

Uwagi dotyczące języka

Ten temat zawiera informacje pomocne podczas korzystania z interfejsu MQI w języku programowania RPG.

Niektóre z tych zagadnień dotyczących języka to:

- "Pliki COPY" na stronie 1037
- "Wywołania" na stronie 1039
- "Parametry wywołania" na stronie 1039
- "Struktury" na stronie 1040
- "Stałe nazwane" na stronie 1040
- "Procedury MQI" na stronie 1040
- "Uwagi dotyczące wątków" na stronie 1040
- "Kontrola transakcji" na stronie 1041
- "Kodowanie powiązanych wywołań" na stronie 1041
- "Konwencje notacyjne" na stronie 1042

Pliki COPY

Dostępne są różne pliki COPY, które pomagają w pisaniu aplikacji RPG korzystających z kolejkowania komunikatów. Istnieją trzy zestawy plików COPY:

- Pliki COPY o nazwach kończących się na *G* są przeznaczone dla programów używających konsolidacji statycznej. Te pliki są inicjowane z wyjątkami określonymi w sekcji "Struktury" na stronie 1040.
- Pliki COPY o nazwach kończących się literą *H* są przeznaczone dla programów, które korzystają z połączenia statycznego, ale **nie** są inicjowane.
- Pliki COPY o nazwach kończących się na *R* są przeznaczone dla programów używających konsolidacji dynamicznej. Te pliki są inicjowane z wyjątkami określonymi w sekcji "Struktury" na stronie 1040.

Zbiory COPY znajdują się w bibliotece QRPGLESRC w bibliotece QMQM.

Dla każdego zestawu plików COPY istnieją dwa pliki zawierające stałe nazwane i jeden plik dla każdej struktury. Tabela 680 na stronie 1038zawiera podsumowanie plików COPY.

Tabela 680. Pliki COPY języka RPG

Nazwa pliku (połączenie statyczne, zainicjowane, CMQ* G)	**Nazwa pliku (połączenie statyczne, niezainicjowane, CMQ* H)**	**Nazwa pliku (połączenie dynamiczne, zainicjowane, CMQ* R)**	**Spis treści**
KMQBOG	CMQBOH	-	Struktura opcji początku
CMQCDG	CMQCDH	CMQCDR	Struktura definicji kanału
CMQCFBFG	CMQCFBFH	-	Parametr filtru bitów PCF
Komenda CMQCFG	-	-	Stałe dla PCF i zdarzeń
CMQCFBSG,	CMQCFBSH	-	Łańcuch bajtowy PCF
CMQCFGRG	CMQCFGRH	-	Parametr grupy PCF
CMQCFIFG	CMQCFIFH	-	Parametr filtru liczb całkowitych PCF
CMQCFHG	CMQCFHH	-	Nagłówek PCF
CMQCFILG	CMQCFILH	-	Struktura parametru listy liczb całkowitych PCF
CMQCFING,	CMQCFINH	-	Struktura parametru liczby całkowitej PCF
CMQCFSFG	CMQCFSFH	-	Parametr filtru łańcucha PCF
CMQCFSLG	CMQCFSLH	-	Struktura parametru listy łańcuchów PCF
CMQCFSTG	CMQCFSTH	-	Struktura parametru łańcucha PCF
CMQCFXLG,	CMQCFXLH	-	Skrócona nazwa PCF dla CFIL64
CMQCFXNG	CMQCFXNH	-	Skrócona nazwa PCF dla CFIN64
CMQCIHG,	CMQCIHH	-	Struktura nagłówka informacji CICS
CMQCNOG	CMQCNOH	-	Struktura opcji połączenia
CMQCSPG	CMQCSPH	-	Parametry zabezpieczeń
CMQCXPG	CMQCXPH	CMQCXPR	Struktura parametru wyjścia kanału
CMQDHG,	CMQDHH	CMQDHR	Struktura nagłówka dystrybucji
CMQDLHG	CMQDLHH	CMQDLHR	Struktura nagłówka niedostarczonego komunikatu
CMQDXPG	CMQDXPH	CMQDXPR	Struktura parametru wyjścia konwersji danych
CMQEPHG	CMQEPHH	-	Wbudowana struktura nagłówka PCF
KMQG	-	CMQR	Stałe nazwane dla głównego interfejsu MQI
CMQGMOG	CMQGMOH	CMQGMOR	Pobierz strukturę opcji komunikatu
CMQIIHG	CMQIIHH	CMQIIHR	Struktura nagłówka informacji IMS
CMQMDEG	CMQMDEH	CMQMDER	Struktura rozszerzenia deskryptora komunikatu

Tabela 680. Pliki COPY języka RPG (kontynuacja)			
Nazwa pliku (połączenie statyczne, zainicjowane, CMQ* G)	**Nazwa pliku (połączenie statyczne, niezainicjowane, CMQ* H)**	**Nazwa pliku (połączenie dynamiczne, zainicjowane, CMQ* R)**	**Spis treści**
CMQMDG	CMQMDH	CMQMDR	Struktura deskryptora komunikatu
CMQMD1G	CMQMD1H	CMQMD1R	Struktura deskryptora komunikatu, wersja 1
CMQMD2G	CMQMD2H	-	Struktura deskryptora komunikatu, wersja 2
CMQODG	CMQODH	CMQODR	Struktura deskryptora obiektu
KMQORG	KMQORH	KMQORR	Struktura rekordu obiektu
CMQPMOG	CMQPMOH	CMQPMOR	Struktura opcji umieszczania komunikatu
CMQPSG	-	-	Stałe dla publikowania/subskrybowania
CMQRFHG	CMQRFHH	-	Reguły i struktura nagłówka formatowania
CMQRFH2G	CMQRFH2H	-	Struktura nagłówka 2 reguł i formatowania
CMQRMHG	CMQRMHH	CMQRMHR	Struktura nagłówka komunikatu odwołania
CMQRRG	CMQRRH	CMQRRR	Struktura rekordu odpowiedzi
CMQTMCG	CMQTMCH	CMQTMCR	Struktura komunikatu wyzwalacza (format znakowy)
CMQTMC2G	CMQTMC2H	CMQTMC2R	Struktura komunikatu wyzwalacza (format znakowy) wersja 2
CMQTMG	CMQTMH	CMQTMR	Struktura komunikatu wyzwalacza
CMQWIHG	CMQWIHH	-	Struktura nagłówka informacji o pracy
CMQXG	-	KMQXR	Stałe nazwane dla wyjścia konwersji danych
CMQXQHG	CMQXQHH	KMQXQHR	Struktura nagłówka kolejki transmisji

Wywołania

Wywołania są opisywane przy użyciu ich indywidualnych nazw.

Parametry wywołania

Niektóre parametry przekazane do interfejsu MQI mogą mieć więcej niż jedną funkcję współbieżną. Dzieje się tak dlatego, że przekazana wartość całkowita jest często testowana na ustawieniu poszczególnych bitów w polu, a nie na jego wartości całkowitej. Pozwala to na 'dodanie' kilku funkcji i przekazanie ich jako pojedynczego parametru.

Struktury

Wszystkie struktury IBM MQ są definiowane z wartościami początkowymi dla pól, z następującymi wyjątkami:

- Dowolna struktura z przyrostkiem H.
- MQTMC
- MQTMC2

Te wartości początkowe są zdefiniowane w odpowiedniej tabeli dla każdej struktury.

Deklaracje struktur nie zawierają instrukcji DS . Dzięki temu aplikacja może zadeklarować pojedynczą strukturę danych lub strukturę danych o wielu wystąpieniach, kodując instrukcję DS , a następnie używając instrukcji /COPY do skopiowania pozostałej części deklaracji:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7
D* Declare an MQMD data structure with 5 occurrences
DMYMD        DS                          5
D/COPY CMQMDR
```

Stałe nazwane

Istnieje wiele wartości całkowitych i znakowych, które umożliwiają wymianę danych między aplikacją i menedżerem kolejek. Aby ułatwić bardziej czytelne i spójne podejście do korzystania z tych wartości, zdefiniowano dla nich stałe nazwane. Można użyć tych stałych nazwanych, a nie wartości, które reprezentują, ponieważ poprawia to czytelność kodu źródłowego programu.

Jeśli plik COPY CMQG jest dołączony do programu w celu zdefiniowania stałych, kompilator RPG zgłosi wiele komunikatów o poziomie ważności zerowym dla stałych, które nie są używane przez program. Komunikaty te są łagodne i można je bezpiecznie zignorować.

Procedury MQI

Jeśli używane są wywołania skonsolidowane ILE, podczas tworzenia programu należy wykonać powiązanie z procedurami MQI. Procedury te są odpowiednio eksportowane z następujących programów usługowych:

QMQM/LIBMQM,
Ten program usługowy zawiera powiązania jednowątkowe dla wersji 5.1 i nowszych. W poniższej sekcji opisano specjalne zagadnienia związane z pisaniem aplikacji wielowątkowych.

QMQM/LIBMQM_R
Ten program usługowy zawiera wielowątkowe powiązania dla wersji 5.1 i nowszych. W poniższej sekcji opisano specjalne zagadnienia związane z pisaniem aplikacji wielowątkowych.

QMQM/LIBMQIC,
Ten program usługowy służy do wiązania niewielowątkowych aplikacji klienckich.

QMQM/LIBMQIC_R
Ten program usługowy służy do wiązania wielowątkowych aplikacji klienckich.

Do tworzenia programów służy komenda CRTPGM . Na przykład poniższa komenda tworzy program jednowątkowy, który używa skonsolidowanych wywołań ILE:

```
CRTPGM PGM(MYPROGRAM) BNDSRVPGM(QMQM/LIBMQM)
```

Uwagi dotyczące wątków

Kompilator języka RPG używany w systemie IBM i jest częścią programów WebSphere Development Toolset i WebSphere Development Studio for IBM i i jest znany jako kompilator języka ILE RPG IV.

Ogólnie rzecz biorąc, programy RPG nie powinny używać wielowątkowych programów usługowych. Wyjątki to programy RPG utworzone za pomocą kompilatora ILE RPG IV i zawierające słowo kluczowe THREAD(*SERIALIZE) w specyfikacji elementu sterującego. Jednak nawet jeśli programy te są wątkowo bezpieczne, należy dokładnie rozważyć ogólny projekt aplikacji, ponieważ THREAD(*SERIALIZE) wymusza serializację procedur RPG na poziomie modułu, co może mieć negatywny wpływ na ogólną wydajność.

Jeśli programy RPG są używane jako wyjścia konwersji danych, muszą być wątkowo bezpieczne i powinny być rekompilowane przy użyciu kompilatora ILE RPG w wersji 4.4 lub nowszej, z wartością THREAD(*SERIALIZE) określoną w specyfikacji sterowania.

Więcej informacji na temat wielowątkowości zawiera publikacja *IBM i IBM MQ Development Studio: ILE RPG Reference*oraz podręcznik *IBM i IBM MQ Development Studio: ILE RPG Programmer's Guide*.

Kontrola transakcji

Funkcje punktu synchronizacji MQI MQCMIT i MQBACK są dostępne dla programów ILE RPG działających w trybie normalnym. Wywołania te umożliwiają programowi zatwierdzanie i wycofywanie zmian w zasobach MQ .

Kodowanie powiązanych wywołań

Procedury MQI środowiska ILE są wymienione w sekcji Tabela 681 na stronie 1041.

Tabela 681. Wywołania skonsolidowane ILE RPG obsługiwane przez każdy program usługowy

Nazwa połączenia	LIBMQM i LIBMQM_R	LIBMQIC i LIBMQIC_R
MQBACK	Y	Y
MQBEGIN	Y	Y
MQCMIT	Y	Y
MQCLOSE	Y	Y
ZMQCONN	Y	Y
MQCONNX	Y	Y
MQDISC	Y	Y
MQGET	Y	Y
MQINQ	Y	Y
MQOPEN	Y	Y
MQPUT	Y	Y
MQPUT1	Y	Y
MQSET	Y	Y
MQXCNPPW	Y	Y

Aby użyć tych procedur, należy wykonać następujące czynności:

1. Zdefiniuj procedury zewnętrzne w specyfikacji D. Są one dostępne w podzbiorze zbioru COPY CMQG zawierającym stałe nazwane.
2. Użyj kodu operacji CALLP, aby wywołać procedurę wraz z jej parametrami.

Na przykład wywołanie MQOPEN wymaga uwzględnienia następującego kodu:

```
D*****************************************************************
D**  MQOPEN Call -- Open Object  (From COPY file CMQG)           **
```

```
D*****************************************************************
D*
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQOPEN           PR                  EXTPROC('MQOPEN')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object descriptor
D OBJDSC                        224A
D* Options that control the action of MQOPEN
D OPTS                           10I 0 VALUE
D* Object handle
D HOBJ                           10I 0
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
D*
```

Aby wywołać procedurę, po zainicjowaniu różnych parametrów potrzebny jest następujący kod:

```
...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+....8
    C                   CALLP     MQOPEN(HCONN : MQOD : OPTS : HOBJ :
    C                             CMPCOD : REASON)
```

W tym przypadku struktura MQOD jest definiowana za pomocą elementu COPY CMQODG, który dzieli ją na komponenty.

Konwencje notacyjne

Te ostatnie tematy w tej sekcji pokazują, w jaki sposób:

- Wywołania powinny być wywoływane
- Parametry powinny być deklarowane
- Należy zadeklarować różne typy danych

W wielu przypadkach parametry są tablicami lub łańcuchami znaków o nieustalonym rozmiarze.
W przypadku tych wartości do reprezentowania stałej liczbowej używana jest mała litera "n" . Jeśli deklaracja dla tego parametru jest zakodowana, wartość "n" musi zostać zastąpiona wymaganą wartością liczbową.

IBM i MQAIR (rekord informacji uwierzytelniającej) w systemie IBM i

Struktura MQAIR reprezentuje rekord informacji uwierzytelniającej.

Przegląd

Cel: Struktura MQAIR umożliwia aplikacji działającej jako klient IBM MQ określenie informacji o elemencie uwierzytelniającym, który ma być używany na potrzeby połączenia klienta. Struktura jest parametrem wejściowym wywołania MQCONNX.

Zestaw znaków i kodowanie: Dane w MQAIR muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT.

- “Pola” na stronie 1042
- “Wartości początkowe” na stronie 1044
- “Deklaracja RPG” na stronie 1045

Pola

Struktura MQAIR zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

AICN (10-cyfrowa liczba całkowita ze znakiem)

Jest to nazwa hosta lub adres sieciowy hosta, na którym działa serwer LDAP. Po tym numerze można podać opcjonalny numer portu ujęty w nawiasy.

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2387.

Domyślny numer portu to 389.

Jest to pole wejściowe. Długość tego pola jest określona przez LNAICN. Wartość początkowa tego pola jest pusta.

AITYP (10-cyfrowa liczba całkowita ze znakiem)

Jest to typ informacji uwierzytelniających zawartych w rekordzie.

Wartość musi być następująca:

AITLDP
Odwołanie certyfikatu przy użyciu serwera LDAP.

Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2386.

Jest to pole wejściowe. Wartością początkową tego pola jest AITLDP.

AIPW (10-cyfrowa liczba całkowita ze znakiem)

Jest to hasło wymagane do uzyskania dostępu do serwera CRL LDAP.

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Jeśli serwer LDAP nie wymaga hasła lub jeśli pominięto nazwę użytkownika LDAP, parametr *AIPW* musi mieć wartość null lub być pusty. Jeśli nazwa użytkownika LDAP zostanie pominięta, a parametr *AIPW* nie będzie pusty, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2390.

Jest to pole wejściowe. Długość tego pola jest określona przez LNLDPW. Początkowa wartość tego pola zawiera puste znaki.

AILUL (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość w bajtach nazwy użytkownika LDAP adresowanej przez pole *AILUP* lub *AILUO* . Wartość musi należeć do zakresu od 0 do LNDISN. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2389.

Jeśli używany serwer LDAP nie wymaga nazwy użytkownika, należy ustawić w tym polu wartość zero.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

AILUO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach nazwy użytkownika LDAP od początku struktury MQAIR.

Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli parametr *LDAPUserNameLength* ma wartość zero.

Do określenia nazwy użytkownika LDAP można użyć wartości *LDAPUserNamePtr* lub *LDAPUserNameOffset* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola *LDAPUserNamePtr* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

AILUP (10-cyfrowa liczba całkowita ze znakiem)

Jest to nazwa użytkownika LDAP.

Składa się ona z nazwy wyróżniającej użytkownika, który próbuje uzyskać dostęp do serwera CRL LDAP. Jeśli wartość jest krótsza niż długość określona przez parametr *AILUL*, należy ją zakończyć znakiem o kodzie zero lub dopełnić spacjami do długości *AILUL*. Pole jest ignorowane, jeśli parametr *AILUL* ma wartość zero.

Nazwę użytkownika LDAP można podać na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *AILUP*

  W takim przypadku aplikacja może zadeklarować łańcuch, który jest oddzielony od struktury MQAIR, i ustawić parametr *AILUP* na adres łańcucha.

  Należy rozważyć użycie języka *AILUP* dla języków programowania obsługujących typ danych wskaźnika w sposób, który jest przenośny w różnych środowiskach (na przykład w języku programowania C).

- Używając pola przesunięcia *AILUO*

  W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą strukturę MQSCO, po której następuje tablica rekordów MQAIR, po której następuje łańcuch nazwy użytkownika LDAP, i ustawić wartość *AILUO* na przesunięcie odpowiedniego łańcucha nazwy od początku struktury MQAIR. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być ustawiona w MQLONG (najbardziej restrykcyjnym językiem programowania jest COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

  Należy rozważyć użycie języka *AILUO* dla języków programowania, które nie obsługują typu danych wskaźnika lub implementują typ danych wskaźnika w sposób, który może nie być przenośny w różnych środowiskach (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki należy używać tylko jednej z następujących metod: *AILUP* i *AILUO* . Wywołanie nie powiodło się z kodem przyczyny RC2388.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

AISID (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

AISIDV
Identyfikator rekordu informacji uwierzytelniającej.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest AISIDV.

AIVER (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

AIVER1
Version-1 rekord informacji uwierzytelniającej.

Następująca stała określa numer wersji bieżącej:

AIRVERC
Bieżąca wersja rekordu informacji uwierzytelniającej.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest AIVER1.

Wartości początkowe

Tabela 682. Pola w MQAIR dla MQAIR

Nazwa pola	Nazwa stałej	Wartość stałej
AISID	AISIDV	'AIR¬'
AIVER	AVERC	1
AITYP	AITLDP	1

Tabela 682. Pola w MQAIR dla MQAIR (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
AICN	Brak	Pusty łańcuch lub odstępy
AILUP	Brak	Pusty wskaźnik lub puste bajty
AILUO	Brak	0
AILUL	Brak	0
AIPW	Brak	Pusty łańcuch lub odstępy

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQAIR Structure
D*
D* Structure identifier
D  AISID                  1      4    INZ('AIR ')
D* Structure version number
D  AIVER                  5      8I 0 INZ(1)
D* Type of authentication information
D  AITYP                  9     12I 0 INZ(1)
D* Connection name of CRL LDAP server
D  AICN                  13    276    INZ
D* Address of LDAP user name
D  AILUP                277    292*   INZ(*NULL)
D* Offset of LDAP user name from start of MQAIR structure
D  AILUO                293    296I 0 INZ(0)
D* Length of LDAP user name
D  AILUL                297    300I 0 INZ(0)
D* Password to access LDAP server
D  AIPW                 301    332    INZ
```

IBM i MQBMHO (opcje od buforu do uchwytu komunikatu) w systemie IBM i

Struktura definiująca opcje obsługi buforu komunikatów.

Przegląd

Cel: Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia uchwytów komunikatów z buforów. Struktura jest parametrem wejściowym wywołania MQBUFMH.

Zestaw znaków i kodowanie: dane w MQBMHO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1045
- "Wartości początkowe" na stronie 1046
- "Deklaracja RPG" na stronie 1046

Pola

Struktura MQBMHO zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

BMSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura uchwytu buforu do komunikatu-pole StrucId .

Jest to identyfikator struktury. Wartość musi być następująca:

BMSIDV
Identyfikator struktury uchwytu buforu do komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest BMSIDV.

BMVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura uchwytu buforu do komunikatu-pole Wersja.

Jest to numer wersji struktury. Wartość musi być następująca:

BMVER1
Numer wersji struktury uchwytu buforu do komunikatu.

Następująca stała określa numer wersji bieżącej:

BMVERVC
Bieżąca wersja struktury uchwytu buforu do komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest BMVER1.

BMOPT (10-cyfrowa liczba całkowita ze znakiem)

Struktura uchwytu buforu do komunikatu-pole Opcje.

Możliwe wartości:

BMDLPR
Właściwości, które są dodawane do uchwytu komunikatu, są usuwane z buforu. Jeśli wywołanie nie powiedzie się, nie zostaną usunięte żadne właściwości.

Opcje domyślne: Jeśli opisywana opcja nie jest potrzebna, należy użyć następującej opcji:

BMNONE
Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest BMDLPR.

Wartości początkowe

Tabela 683. Pola w MQBMHO

Nazwa pola	Nazwa stałej	Wartość stałej
BMSID	BMSIDV	'BMHO'
BMVER	BMVER1	1
BMOPT	BMNONE	0

Deklaracja RPG

```
D* MQBMHO Structure
D*
D*
D* Structure identifier
D  BMSID                  1      4    INZ('BMHO')
D*
D* Structure version number
D  BMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQBUFMH
D  BMOPT                  9     12I 0 INZ(1)
```

IBM i MQBO (opcje początku) w systemie IBM i

Struktura MQBO umożliwia aplikacji określenie opcji związanych z tworzeniem jednostki pracy.

Przegląd

Cel: Struktura jest parametrem wejścia/wyjścia w wywołaniu komendy MQBEGIN.

Zestaw znaków i kodowanie: Dane w obiekcie MQBO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez translację ENNAT.

- "Pola" na stronie 1047
- "Wartości początkowe" na stronie 1047
- "Deklaracja RPG" na stronie 1048

Pola

Struktura MQBO zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

BOOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje sterujące działaniem komendy MQBEGIN.

Wartość musi być następująca:

BONONE.
Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest BONONE.

BOSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

BOSIDV
Identyfikator struktury opcji początku.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest BOSIDV.

BOVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

BOVER1
Numer wersji dla struktury opcji początku.

Następująca stała określa numer wersji bieżącej:

BOVERC
Bieżąca wersja struktury opcji rozpoczęcia.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest BOVER1.

Wartości początkowe

Tabela 684. Pola w obiekcie MQBO

Nazwa pola	Nazwa stałej	Wartość stałej
BOSID	BOSIDV	'BO¬¬'
BOVER	BOVER1	1

Tabela 684. Pola w obiekcie MQBO (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
BOOPT	BONONE.	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQBO Structure
D*
D* Structure identifier
D  BOSID                  1      4    INZ('BO  ')
D* Structure version number
D  BOVER                  5      8I 0 INZ(1)
D* Options that control the action of MQBEGIN
D  BOOPT                  9     12I 0 INZ(0)
```

IBM i MQCBC (kontekst wywołania zwrotnego) w systemie IBM i

Struktura opisująca procedurę zwrotną.

Przegląd

Przeznaczenie
Struktura MQCBC służy do określania informacji o kontekście przekazywanych do funkcji zwrotnej.

Struktura jest parametrem wejścia/wyjścia w wywołaniu procedury konsumenta komunikatów.

Wersja
Bieżąca wersja MQCBC to CBCV2.

Zestaw znaków i kodowanie
Dane w tabeli MQCBC znajdują się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura jest w zestawie znaków i kodowaniu klienta.

- “Pola” na stronie 1048
- “Wartości początkowe” na stronie 1054
- “Deklaracja RPG” na stronie 1054

Pola

Struktura MQCBC zawiera następujące pola; pola są opisane w porządku alfabetycznym:

CBCBUFFLEN (10-cyfrowa liczba całkowita ze znakiem)

Bufor może być większy niż zarówno wartość MaxMsgLength zdefiniowana dla konsumenta, jak i wartość ReturnedLength w obiekcie MQGMO.

Struktura kontekstu wywołania zwrotnego-pole BufferLength .

Jest to długość (w bajtach) buforu komunikatów, który został przekazany do tej funkcji.

Rzeczywista długość komunikatu jest podawana w polu DataLength .

Aplikacja może używać całego buforu do własnych celów w czasie trwania funkcji zwrotnej.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi wyjątku.

CBCCALLBA (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CallbackArea .

Jest to pole dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest ono przekazywane bez zmian z pola CBDCALLBA w strukturze MQCBD, która jest parametrem wywołania MQCB używanym do definiowania funkcji zwrotnej.

Zmiany wprowadzone w pliku *CBCCALLBA* są zachowywane w wywołaniach funkcji zwrotnej dla *CBCHOBJ*. To pole nie jest współużytkowane z funkcjami zwrotnymi dla innych uchwytów.

Jest to pole wejściowe/wyjściowe funkcji zwrotnej. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

CBCCALLT (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CallType .

Pole zawierające informacje o przyczynie wywołania tej funkcji. Zdefiniowane są następujące typy wywołań.

Typy wywołań dostarczania komunikatów: te typy wywołań zawierają informacje o komunikacie. Parametry **CBCLEN** i **CBCBUFFLEN** są poprawne dla tych typów wywołań.

CBCTMR
Funkcja konsumenta komunikatów została wywołana z komunikatem, który został destrukcyjnie usunięty z uchwytu obiektu.

Jeśli wartością *CBCCC* jest CCWARN, wartością pola *Reason* jest RC2079 lub jeden z kodów wskazujących na problem z konwersją danych.

CBCTMN (SIEĆ)
Funkcja konsumenta komunikatów została wywołana z komunikatem, który nie został jeszcze destrukcyjnie usunięty z uchwytu obiektu. Komunikat można destrukcyjnie usunąć z uchwytu obiektu za pomocą *MsgToken*.

Komunikat mógł nie zostać usunięty, ponieważ:

- Opcje MQGMO zażądały operacji przeglądania, GMBR*
- Komunikat jest większy niż dostępny bufor, a opcje MQGMO nie określają wartości gmatm

Jeśli wartością *CBCCC* jest CCWARN, wartością pola *Reason* jest RC2080 lub jeden z kodów wskazujących na problem z konwersją danych.

Typy wywołań sterujących wywołania zwrotnego: te typy wywołań zawierają informacje o sterowaniu wywołaniem zwrotnym i nie zawierają szczegółów dotyczących komunikatu. Te typy wywołań są żądane przy użyciu opcji CBDOPT w strukturze MQCBD.

Parametry **CBCLEN** i **CBCBUFFLEN** nie są poprawne dla tych typów wywołań.

CCBCTRC
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania początkowej konfiguracji.

Funkcja zwrotna jest wywoływana natychmiast po zarejestrowaniu wywołania zwrotnego, czyli po powrocie z wywołania MQCB z użyciem wartości pola *Operation* w CBREG.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i dla procedur obsługi zdarzeń.

Na żądanie jest to pierwsze wywołanie funkcji zwrotnej.

Wartością pola *CBCREA* jest RCNONE.

CBCTSC
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania pewnych czynności konfiguracyjnych podczas uruchamiania, na przykład przywrócenie zasobów, które zostały wyczyszczone podczas poprzedniego zatrzymania.

Funkcja zwrotna jest wywoływana po uruchomieniu połączenia przy użyciu protokołu CTLSR lub CTLSW.

Jeśli funkcja zwrotna jest zarejestrowana w innej funkcji zwrotnej, ten typ wywołania jest wywoływany po powrocie z wywołania zwrotnego.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartością pola *CBCREA* jest RCNONE.

CBCTTC,
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania procedury czyszczącej, gdy zostanie ona zatrzymana na jakiś czas, na przykład w celu wyczyszczenia dodatkowych zasobów, które zostały uzyskane podczas odbierania komunikatów.

Funkcja zwrotna jest wywoływana, gdy wywołanie MQCTL jest wykonywane przy użyciu wartości w polu *Operation* procesora CTLSP.

Ten typ wywołania jest używany tylko dla konsumentów komunikatów.

Wartość w polu *CBCREA* jest ustawiona w celu wskazania przyczyny zatrzymania.

CBCTDC
Celem tego typu wywołania jest umożliwienie funkcji zwrotnej wykonania końcowego czyszczenia po zakończeniu procesu konsumowania. Funkcja zwrotna jest wywoływana, gdy:

- Funkcja zwrotna jest wyrejestrowywana przy użyciu wywołania MQCB z BCUNR.
- Kolejka jest zamknięta, co powoduje niejawne wyrejestrowanie. W tym przypadku funkcja zwrotna jest przekazywana jako uchwyt obiektu HOUNUH.
- Wywołanie MQDISC zostało zakończone-spowodowało niejawne zamknięcie i wyrejestrowanie. W takim przypadku połączenie nie jest rozłączane natychmiast, a żadna transakcja w toku nie jest jeszcze zatwierdzona.

Jeśli dowolne z tych działań zostanie wykonane wewnątrz samej funkcji zwrotnej, działanie zostanie wywołane po powrocie z wywołania zwrotnego.

Ten typ wywołania jest używany zarówno dla konsumentów komunikatów, jak i dla procedur obsługi zdarzeń.

Na żądanie jest to ostatnie wywołanie funkcji zwrotnej.

Wartość w polu *CBCREA* jest ustawiona w celu wskazania przyczyny zatrzymania.

CBCTEC,
Funkcja procedury obsługi zdarzeń

Funkcja procedury obsługi zdarzeń została wywołana bez komunikatu, gdy:

- Wywołanie MQCTL jest wykonywane z wartością pola *Operation* w CTLSP lub
- Menedżer kolejek lub połączenie zatrzymuje się lub wycisza.

Tego wywołania można użyć do podjęcia odpowiednich działań dla wszystkich funkcji zwrotnych.

- **Funkcja konsumenta komunikatów**

  Funkcja konsumenta komunikatów została wywołana bez komunikatu w przypadku wykrycia błędu (*CBCCC* = CCFAIL), który jest specyficzny dla uchwytu obiektu, na przykład *CBCREA* code = RC2016 .

Wartość w polu *CBCREA* jest ustawiona w celu wskazania przyczyny wywołania.

Jest to pole wejściowe. CBCTMR i CMCTMN mają zastosowanie tylko do funkcji konsumenta komunikatów.

CBCCC (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CompCode .

Jest to kod zakończenia. Wskazuje, czy wystąpiły problemy podczas odbierania komunikatu. Jest to jedna z następujących opcji:

CKOK
Pomyślne zakończenie

CWARN
Ostrzeżenie (częściowe zakończenie)

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się

Jest to pole wejściowe. Wartością początkową tego pola jest CCOK.

CBCCONNAREA (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole ConnectionArea .

Jest to pole dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest ono przekazywane bez zmian z pola ConnectionArea w strukturze MQCTLO, które jest parametrem wywołania MQCTL używanym do sterowania funkcją zwrotną.

Wszelkie zmiany wprowadzone w tym polu przez funkcje zwrotne są zachowywane w wywołaniach funkcji zwrotnej. Ten obszar może być używany do przekazywania informacji, które mają być współużytkowane przez wszystkie funkcje zwrotne. W przeciwieństwie do obszaru *CallbackArea*, ten obszar jest wspólny dla wszystkich wywołań zwrotnych uchwytu połączenia.

Jest to pole wejściowe i wyjściowe. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

CBCLEN (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość (w bajtach) danych aplikacji w komunikacie. Jeśli wartość wynosi zero, oznacza to, że komunikat nie zawiera danych aplikacji.

Pole CBCLEN zawiera długość komunikatu, ale niekoniecznie długość danych komunikatu przekazanych do konsumenta. Możliwe, że komunikat został obcięty. Użyj pola GMRL w MQGMO, aby określić ilość danych, które zostały przekazane do konsumenta.

Jeśli kod przyczyny wskazuje, że komunikat został obcięty, można użyć pola CBCLEN, aby określić wielkość rzeczywistego komunikatu. Umożliwia to określenie wielkości buforu wymaganego do przechowywania danych komunikatu, a następnie wywołanie obiektu MQCB w celu zaktualizowania pola CBDMML w obiekcie MQCBD przy użyciu odpowiedniej wartości.

Jeśli podano opcję GMCONV, przekształcony komunikat może być większy niż wartość zwrócona dla DataLength. W takich przypadkach aplikacja prawdopodobnie musi wydać wywołanie MQCB w celu zaktualizowania zmiennej CBDMML w MQCBD tak, aby była większa niż wartość zwrócona przez menedżer kolejek dla parametru DataLength.

Aby uniknąć problemów z obcinaniem komunikatów, należy określić wartość MaxMsgLength jako CBDFM. Powoduje to, że menedżer kolejek przydziela bufor dla pełnej długości komunikatu po konwersji danych. Należy jednak pamiętać, że nawet jeśli ta opcja jest określona, nadal możliwe jest, że nie jest dostępna wystarczająca ilość pamięci do poprawnego przetworzenia żądania. Aplikacje powinny zawsze sprawdzać zwrócony kod przyczyny. Jeśli na przykład nie można przydzielić wystarczającej ilości pamięci do przekształcenia komunikatu, komunikaty są zwracane do aplikacji bez konwersji.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

CBCFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi zawierające informacje o tym konsumencie.

Zdefiniowana jest następująca opcja:

CBCFBE,
Ta opcja może zostać zwrócona, jeśli poprzednie wywołanie MQCLOSE z opcją COQSC nie powiodło się z kodem przyczyny RC2458.

Ten kod wskazuje, że jest zwracany ostatni komunikat odczytu z wyprzedzeniem i że bufor jest teraz pusty. Jeśli aplikacja wyśle kolejne wywołanie MQCLOSE z użyciem opcji COQSC, zakończy się powodzeniem.

Należy zauważyć, że nie ma gwarancji, że aplikacja otrzyma komunikat z tym zestawem flag, ponieważ w buforze odczytu z wyprzedzeniem mogą nadal znajdować się komunikaty, które nie są zgodne z bieżącymi kryteriami wyboru. W tym przypadku funkcja konsumenta jest wywoływana z kodem przyczyny RC2019 .

Jeśli bufor odczytu z wyprzedzeniem jest pusty, konsument jest wywoływany z flagą CBCFBE i kodem przyczyny RC2518.

Jest to pole wejściowe dla funkcji konsumenta komunikatów. Nie jest ono istotne dla funkcji procedury obsługi zdarzeń.

CBCHOBJ (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole CBCHOBJ.

W przypadku wywołania do konsumenta komunikatów jest to uchwyt obiektu związanego z konsumentem komunikatów.

W przypadku procedury obsługi zdarzeń ta wartość to HONONE

Aplikacja może użyć tego uchwytu i znacznika komunikatu w bloku Opcje pobierania komunikatu, aby pobrać komunikat, jeśli komunikat nie został usunięty z kolejki.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest HOUNUH

CBCRCD (10-cyfrowa liczba całkowita ze znakiem)

Wartość **CBCRCD** wskazuje, jak długo menedżer kolejek oczekuje przed podjęciem próby ponownego nawiązania połączenia. Pole może być modyfikowane przez procedurę obsługi zdarzeń w celu zmiany opóźnienia lub zatrzymania ponownego połączenia.

Pola **CBCRCD** należy używać tylko wtedy, gdy wartością pola **Reason** w kontekście wywołania zwrotnego jest RC2545.

We wpisie do procedury obsługi zdarzeń wartość **CBCRCD** jest liczbą milisekund, przez które menedżer kolejek będzie oczekiwał przed podjęciem próby ponownego nawiązania połączenia. Tabela 685 na stronie 1052 zawiera listę wartości, które można ustawić w celu zmodyfikowania zachowania menedżera kolejek w przypadku powrotu z procedury obsługi zdarzeń.

*Tabela 685. **CBCRCD** wartości*

Wartość	Opis
-1	Nie wykonuj więcej prób ponownego połączenia. Do aplikacji jest zwracany błąd.
0	Spróbuj połączyć się ponownie natychmiast.
>0	Przed ponowną próbą nawiązania połączenia odczekaj ten czas (w milisekundach).

CBCREA (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole Przyczyna.

Jest to kod przyczyny określający *CBCCC*

Jest to pole wejściowe. Wartością początkową tego pola jest RCNONE.

CBCSTATE (10-cyfrowa liczba całkowita ze znakiem)

Wskazanie stanu bieżącego konsumenta. To pole jest najbardziej wartościowe dla aplikacji, gdy do funkcji konsumenta przekazywany jest niezerowy kod przyczyny.

Tego pola można użyć, aby uprościć programowanie aplikacji, ponieważ nie ma potrzeby kodowania zachowania dla każdego kodu przyczyny.

Jest to pole wejściowe. Wartością początkową tego pola jest CSNONE

Tabela 686. Wartości BCSTATE i wynikowe działania

Stan	**Działanie menedżera kolejek**	**Wartość stałej**
CSNONE Ten kod przyczyny reprezentuje normalne wywołanie bez dodatkowych informacji o przyczynie.	Brak; jest to normalne działanie.	0
CSSUST Te kody przyczyny reprezentują warunki tymczasowe.	Procedura zwrotna jest wywoływana w celu zgłoszenia warunku, a następnie zawieszona. Po pewnym czasie system może ponowić operację, co może doprowadzić do ponownego wystąpienia tego samego warunku.	1
CSSUSU Te kody przyczyny reprezentują warunki, w których wywołanie zwrotne musi działać w celu rozstrzygnięcia warunku.	Konsument jest zawieszony, a procedura wywołania zwrotnego jest wywoływana w celu zgłoszenia warunku. Procedura wywołania zwrotnego powinna rozwiązać warunek, jeśli to możliwe, i albo RESUME, albo zamknąć połączenie.	2
CSSUS Te kody przyczyny reprezentują niepowodzenia, które uniemożliwiają dalsze wywołania zwrotne komunikatów.	Menedżer kolejek automatycznie zawiesza funkcję zwrotną. Jeśli funkcja zwrotna zostanie wznowiona, prawdopodobnie ponownie otrzyma ten sam kod przyczyny.	3
CSSTOP Te kody przyczyny reprezentują koniec wykorzystania komunikatów.	Dostarczone do procedury obsługi wyjątków i do wywołań zwrotnych, które określały CBDTC. Nie można korzystać z kolejnych komunikatów.	4

CBCSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole StrucId .

Jest to identyfikator struktury; wartość musi być następująca:

CBCSI,
Identyfikator struktury kontekstu wywołania zwrotnego.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CBCSI.

CBCVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura kontekstu wywołania zwrotnego-pole Wersja.

Jest to numer wersji struktury; wartość musi być następująca:

CBCV1
Version-1 struktura kontekstu wywołania zwrotnego.

Następująca stała określa numer wersji bieżącej:

CBCCV
Bieżąca wersja struktury kontekstu wywołania zwrotnego.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CBCV1.

Wartości początkowe

Tabela 687. Pola w MQCBC

Nazwa pola	Nazwa stałej	Wartość stałej
CBCSID	CBCSI,	'CBC¬'
CBCVER	CBCV1	1
CBCCALLT	Brak	0
CBCHOBJ	HUNUH	-1
CBCCALLBA	Brak	Pusty wskaźnik lub puste bajty
CBCCONNAREA	Brak	Pusty wskaźnik lub puste bajty
CBCCC	CKOK	0
CBCREA	BRAK RCNONE	0
CBCSTATE	CSNONE	0
CBCLEN	Brak	0
CBCBUFFLEN	Brak	0
CBCFLG	Brak	0
CBCRCD	brak	0

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D* MQCBC Structure
D*
D*
D* Structure identifier
D  CBCSID                 1      4    INZ('CBC ')
D*
D* Structure version number
D  CBCVER                 5      8I 0 INZ(1)
D*
D* Why Function was called
D  CBCCALLT               9     12I 0 INZ(0)
D*
D* Object Handle
D  CBCHOBJ               13     16I 0 INZ(-1)
D*
D* Callback data passed to the function
D  CBCCALLBA             17     32*   INZ(*NULL)
D*
D* MQCTL Data area passed to the function
D  CBCCONNAREA           33     48*   INZ(*NULL)
D*
D* Completion Code
D  CBCCC                 49     52I 0 INZ(0)
D*
```

```
D* Reason Code
D  CBCREA                53     56I 0 INZ(0)
D*
D* Consumer State
D  CBCSTATE              57     60I 0 INZ(0)
D*
D* Message Data Length
D  CBCLEN                61     64I 0 INZ(0)
D*
D* Buffer Length
D  CBCBUFFLEN            65     68I 0 INZ(0)
D*
** Flags containing information about
D* this consumer
D  CBCFLG                69     72I 0 INZ(0)
D* Ver:1 **
D* Number of milliseconds before reconnect attempt
D  CBCRCD                73     76I 0 INZ(0)
D* Ver:2 **
D*
```

IBM i MQCBD (deskryptor Callback) w systemie IBM i

Struktura określająca funkcję zwrotną.

Przegląd

Cel: Struktura MQCBD jest używana do określania funkcji zwrotnej i opcji sterujących jej użyciem przez menedżer kolejek.

Struktura jest parametrem wejściowym wywołania MQCB.

Wersja: Bieżąca wersja MQCBD to CBDV1.

Zestaw znaków i kodowanie: Dane w MQCBD muszą być w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Są one nadawane przez atrybut menedżera kolejek **CodedCharSetId** i ENNAT. Jeśli jednak aplikacja działa jako IBM MQ MQI client, struktura musi być w zestawie znaków i kodowaniu klienta.

- "Pola" na stronie 1055
- "Wartości początkowe" na stronie 1059
- "Deklaracja RPG" na stronie 1059

Pola

Struktura MQCBD zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

CBDCALLBA (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest ono przekazywane bez zmian z pola CBCCALLBA w strukturze MQCBD, która jest parametrem w deklaracji funkcji zwrotnej.

Ta wartość jest używana tylko w przypadku, gdy *Operation* ma wartość CBREG i nie ma obecnie zdefiniowanego wywołania zwrotnego, nie zastępuje poprzedniej definicji.

Jest to pole wejściowe i wyjściowe funkcji zwrotnej. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

CBDCALLBF (10-cyfrowa liczba całkowita ze znakiem)

Funkcja zwrotna jest wywoływana jako wywołanie funkcji.

To pole służy do określania wskaźnika do funkcji zwrotnej.

Należy podać wartość *CallbackFunction* lub *CallbackName*. Jeśli zostaną podane obie te wartości, zostanie zwrócony kod przyczyny RC2486 .

Jeśli nie jest ustawiona opcja *CallbackName* ani *CallbackFunction* , wywołanie kończy się niepowodzeniem z kodem przyczyny RC2486.

Ta opcja nie jest obsługiwana w następujących środowiskach:

- CICSOn z/OS
- Języki programowania i kompilatory, które nie obsługują odwołań do wskaźników funkcji

W takich sytuacjach wywołanie kończy się niepowodzeniem z kodem przyczyny RC2486.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

CBDCALLBN (10-cyfrowa liczba całkowita ze znakiem)

Funkcja zwrotna jest wywoływana jako program dowiązany dynamicznie.

Należy podać wartość *CallbackFunction* lub *CallbackName*. Jeśli zostaną podane obie te wartości, zostanie zwrócony kod przyczyny RC2486 .

Jeśli parametr *CallbackName* lub *CallbackFunction* nie ma wartości true, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2486.

Moduł jest ładowany, gdy rejestrowana jest pierwsza procedura zwrotna, która ma zostać użyta, i rozładowywany, gdy ostatnia procedura zwrotna, która ma go użyć, wyrejestrowuje.

Z wyjątkiem sytuacji opisanych w poniższym tekście, nazwa jest wyrównywana do lewej strony pola, bez odstępów wewnętrznych; sama nazwa jest dopełniana spacjami do długości pola. W poniższych opisach nawiasy kwadratowe ([]) oznaczają informacje opcjonalne:

IBMi
Nazwa wywołania zwrotnego może mieć jeden z następujących formatów:

- Biblioteka "/" Program
- Biblioteka "/" ServiceProgram "("FunctionName")"

Na przykład: `MyLibrary/MyProgram(MyFunction)`.

Nazwą biblioteki może być *LIBL. Nazwy biblioteki i programu są ograniczone do maksymalnie 10 znaków.

AIX and Linux
Nazwa wywołania zwrotnego jest nazwą dynamicznie ładowanego modułu lub biblioteki z dołączonym przyrostkiem nazwy funkcji rezydującej w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu:

```
[path]library(function)
```

Jeśli ścieżka nie zostanie podana, zostanie użyta systemowa ścieżka wyszukiwania.

Długość nazwy nie może przekraczać 128 znaków.

Windows
Nazwa wywołania zwrotnego jest nazwą biblioteki dołączanej dynamicznie z przyrostkiem nazwy funkcji rezydującej w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwę biblioteki można opcjonalnie poprzedzić ścieżką do katalogu i napędem:

```
[d:][path]library(function)
```

Jeśli napęd i ścieżka nie są określone, używana jest systemowa ścieżka wyszukiwania.

Długość nazwy nie może przekraczać 128 znaków.

z/OS
Nazwa wywołania zwrotnego jest nazwą modułu ładującego, która jest poprawna dla specyfikacji parametru EP makra LINK lub LOAD.

Długość nazwy nie może przekraczać 8 znaków.

z/OS CICS

Nazwa wywołania zwrotnego jest nazwą modułu ładującego, która jest poprawna dla specyfikacji parametru PROGRAM makra komendy EXEC CICS LINK.

Długość nazwy nie może przekraczać 8 znaków.

Program można zdefiniować jako zdalny za pomocą opcji REMOTESYTEM zainstalowanej definicji PROGRAM lub za pomocą programu routingu dynamicznego.

Zdalny region CICS musi być połączony z systemem IBM MQ , jeśli program ma używać wywołań funkcji API języka IBM MQ . Należy jednak zauważyć, że pole CBCHOBJ w strukturze MQCBC nie jest poprawne w systemie zdalnym.

Jeśli wystąpi błąd podczas próby załadowania pliku *CallbackName*, do aplikacji zwracany jest jeden z następujących kodów błędu:

- RC2495
- RC2496
- RC2497

W dzienniku błędów zapisywany jest również komunikat zawierający nazwę modułu, dla którego próbowano wykonać ładowanie, oraz kod przyczyny niepowodzenia z systemu operacyjnego.

Jest to pole wejściowe. Wartością początkową tego pola jest łańcuch pusty lub pusty łańcuch.

CBDCALLBT (10-cyfrowa liczba całkowita ze znakiem)

Jest to typ funkcji zwrotnej. Wartość musi być jedną z następujących wartości:

CCBTMC

Definiuje wywołanie zwrotne jako funkcję konsumenta komunikatów.

Funkcja zwrotna konsumenta komunikatów jest wywoływana, gdy komunikat spełniający określone kryteria wyboru jest dostępny w uchwycie obiektu i połączenie jest uruchamiane.

CCBTEH

Definiuje to wywołanie zwrotne jako procedurę zdarzenia asynchronicznego; nie jest sterowane odbieraniem komunikatów dla uchwytu.

Parametr *Hobj* nie jest wymagany w wywołaniu obiektu MQCB definiującym procedurę obsługi zdarzeń i jest ignorowany, jeśli został określony.

Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko konsumenta komunikatów. Funkcja konsumenta jest wywoływana bez komunikatu, gdy wystąpi zdarzenie, na przykład zatrzymanie menedżera kolejek lub połączenia albo wyciszenie. Nie jest ona wywoływana w przypadku warunków specyficznych dla pojedynczego konsumenta komunikatów, na przykład RC2016.

Zdarzenia są dostarczane do aplikacji, niezależnie od tego, czy połączenie jest uruchomione, czy zatrzymane, z wyjątkiem następujących środowisk:

- Środowisko CICS w systemie z/OS
- aplikacje niewielowątkowe

Jeśli program wywołujący nie przekazuje jednej z tych wartości, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2483

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CBTMC.

CBDMML (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość (w bajtach) najdłuższego komunikatu, który można odczytać z uchwytu i przekazać do procedury zwrotnej. Jeśli komunikat ma dłuższą długość, procedura zwrotna odbiera *MaxMsgLength* bajtów komunikatu i kod przyczyny:

- RC2080 lub

- RC2079 , jeśli określono GMATM.

Rzeczywista długość komunikatu jest podawana w polu "CBCLEN (10-cyfrowa liczba całkowita ze znakiem)" na stronie 1051 struktury MQCBC.

Zdefiniowana jest następująca wartość specjalna:

CBDFM,
Długość buforu jest dopasowywana przez system w taki sposób, aby komunikaty były zwracane bez obcinania.

Jeśli ilość dostępnej pamięci jest niewystarczająca do przydzielenia buforu w celu odebrania komunikatu, system wywołuje funkcję zwrotną z kodem przyczyny RC2071 .

Jeśli na przykład żądana jest konwersja danych, a ilość dostępnej pamięci jest niewystarczająca do przekształcenia danych komunikatu, nieprzekształcony komunikat jest przekazywany do funkcji zwrotnej.

Jest to pole wejściowe. Wartością początkową pola *MaxMsgLength* jest CBDFM.

CBDOPT (10-cyfrowa liczba całkowita ze znakiem)

Struktura deskryptora wywołania zwrotnego-pole Opcje.

Można określić jedną lub wszystkie poniższe wartości. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe). Kombinacje, które nie są poprawne, są odnotowywane; wszystkie inne kombinacje są poprawne.

CBDFQ,
Wywołanie MQCB kończy się niepowodzeniem, jeśli menedżer kolejek jest w stanie wyciszania.

W systemie z/OSta opcja wymusza także niepowodzenie wywołania MQCB, jeśli połączenie (dla aplikacji CICS lub IMS) jest w stanie wyciszania.

Określ wartość GMFIQ w opcjach MQGMO przekazanych w wywołaniu MQCB, aby spowodować powiadomienie konsumentów komunikatów podczas wyciszania.

Opcje sterowania: Następujące opcje sterują tym, czy funkcja zwrotna jest wywoływana bez komunikatu, gdy zmienia się stan konsumenta:

CBDRC
Funkcja zwrotna jest wywoływana z typem wywołania CBCTRC

.

CBDSC,
Funkcja zwrotna jest wywoływana z typem wywołania CBCTSC.

CBDTC,
Funkcja zwrotna jest wywoływana z typem wywołania CBCTTC.

CBDDC
Funkcja zwrotna jest wywoływana z typem wywołania CBCTDC.

Więcej informacji na temat tych typów wywołań zawiera sekcja "CBCCALLT (10-cyfrowa liczba całkowita ze znakiem)" na stronie 1049 .

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

CBDNO
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

CBDNO jest zdefiniowane jako pomoc dla dokumentacji programu; nie jest zamierzone, aby ta opcja była używana z jakąkolwiek inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może być wykryte.

Jest to pole wejściowe. Wartością początkową pola *Options* jest CBDNO.

CBDSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura deskryptora wywołania zwrotnego-pole StrucId .

Jest to identyfikator struktury; wartość musi być następująca:

CBDSI
Identyfikator struktury deskryptora wywołania zwrotnego.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CBDSI.

CBDVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura deskryptora wywołania zwrotnego-pole Wersja.

Jest to numer wersji struktury; wartość musi być następująca:

CBDV1
Version-1 struktura deskryptora wywołania zwrotnego.

Następująca stała określa numer wersji bieżącej:

CCBDCV
Bieżąca wersja struktury deskryptora wywołania zwrotnego.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CBDV1.

Wartości początkowe

Tabela 688. Pola w MQCBD

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	CBDSI	'CBD¬'
Version	CBDV1	1
CallBackType	CCBTMC	1
Options	CBDNO	0
CallbackArea	Brak	Puste bajty
CallbackFunction	Brak	Puste bajty
CallbackName	Brak	Puste
MaxMsgLength	CBDFM,	-1

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D* MQCBD Structure
D*
D*
D* Structure identifier
D  CBDSID                 1      4    INZ('CBD ')
D*
D* Structure version number
D  CBDVER                 5      8I 0 INZ(1)
D*
D* Callback function type
D  CBDCALLBT              9     12I 0 INZ(1)
D*
** Options controlling message
D* consumption
D  CBDOPT                13     16I 0 INZ(0)
D*
```

```
     D* User data passed to the function
     D  CBDCALLBA             17     32*
     D*
     D* FP: Callback function pointer
     D  CBDCALLBF             33     48*
     D*
     D* Callback name
     D  CBDCALLBN             49    176    INZ('\0')
     D*
     D* Maximum message length
     D  CBDMML               177    180I 0 INZ(-1)
```

IBM i MQCHARV (Variable Length String) w systemie IBM i

Struktura MQCHARV służy do opisu łańcucha o zmiennej długości.

Przegląd

Zestaw znaków i kodowanie: Dane w tabeli MQCHARV muszą być zakodowane w lokalnym menedżerze kolejek, który jest nadawany przez ENNAT i zestaw znaków pola VCHRC w strukturze. Jeśli aplikacja działa jako IBM MQ MQI client, struktura musi być zakodowana w kliencie. Niektóre zestawy znaków mają reprezentację zależną od kodowania. Jeśli VCHRC jest jednym z tych zestawów znaków, użyte kodowanie jest takie samo jak kodowanie innych pól w MQCHARV. Zestaw znaków identyfikowany przez VSCCSID może być zestawem znaków dwubajtowych (DBCS).

Użycie: Struktura MQCHARV adresuje dane, które mogą być nieciągłe w strukturze zawierającej tę strukturę. W celu adresowania tych danych można użyć pól zadeklarowanych za pomocą typu danych wskaźnika.

- "Pola" na stronie 1060
- "Wartości początkowe" na stronie 1061
- "Deklaracja RPG" na stronie 1062
- "Ponowna definicja CSAPL" na stronie 1062

Pola

Struktura MQCHARV zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

VCHRC (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator zestawu znaków łańcucha o zmiennej długości, adresowanego przez pole VCHRP lub VCHRO.

Wartością początkową tego pola jest CSAPL. Wartość ta jest definiowana przez program IBM MQ w celu wskazania, że menedżer kolejek powinien zmienić ją na rzeczywisty identyfikator zestawu znaków menedżera kolejek. Jest tak samo, jak zachowuje się CSQM. W rezultacie wartość CSAPL nigdy nie jest powiązana z łańcuchem o zmiennej długości. Początkową wartość tego pola można zmienić, definiując inną wartość stałej CSAPL dla jednostki kompilacji w sposób odpowiedni dla języka programowania aplikacji.

VCHRL (10-cyfrowa liczba całkowita ze znakiem)

Długość w bajtach łańcucha o zmiennej długości adresowanego przez pole VCHRP lub VCHRO.

Wartością początkową tego pola jest 0. Wartość musi być większa lub równa zero lub następująca wartość specjalna, która jest rozpoznawana:

VSNLT

Jeśli nie określono VSNLT, bajty VCHRL są uwzględniane jako część łańcucha. Jeśli występują znaki o kodzie zero, łańcuch nie jest ograniczany.

Jeśli podano VSNLT, łańcuch jest ograniczony przez pierwszą wartość null napotkaną w łańcuchu. Sama wartość NULL nie jest uwzględniana jako część tego łańcucha.

Uwaga: Znak o kodzie zero używany do zakończenia łańcucha, jeśli określono VSNLT, jest znakiem o kodzie zero z zestawu kodowego określonego przez VCHRC.

Na przykład w UTF-16 (CCSID 1200, 13488 i 17584) jest to 2-bajtowe kodowanie Unicode, w którym wartość null jest reprezentowana przez 16-bitową liczbę zer. W UTF-16 powszechne jest wyszukiwanie pojedynczych bajtów ustawionych na zero, które są częścią znaków (na przykład 7-bitowych znaków ASCII), ale łańcuchy będą zakończone znakiem o kodzie zero tylko wtedy, gdy na granicy parzystego bajtu zostaną znalezione dwa bajty o kodzie zero. Można uzyskać dwa 'zero' bajtów na granicy nieparzystej, gdy są one częścią poprawnych znaków. Na przykład x '01' x '00' x '00' x '30' reprezentuje dwa poprawne znaki Unicode i nie może kończyć łańcucha znakiem null.

VCHRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie (w bajtach) łańcucha o zmiennej długości od początku MQCHARV lub struktury zawierającej ten łańcuch.

Jeśli struktura MQCHARV jest osadzona w innej strukturze, ta wartość jest przesunięciem (w bajtach) łańcucha o zmiennej długości od początku struktury zawierającej tę strukturę MQCHARV. Jeśli struktura MQCHARV nie jest osadzona w innej strukturze, na przykład jeśli jest określona jako parametr w wywołaniu funkcji, przesunięcie jest względne w stosunku do początku struktury MQCHARV.

Przesunięcie może być dodatnie lub ujemne. Do określenia łańcucha o zmiennej długości można użyć pola VCHRP lub VCHRO, ale nie obu.

Wartością początkową tego pola jest 0.

VCHRP (wskaźnik)

Jest to wskaźnik do łańcucha o zmiennej długości.

Do określenia łańcucha o zmiennej długości można użyć pola VCHRP lub VCHRO, ale nie obu.

Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

VCHRS (10-cyfrowa liczba całkowita ze znakiem)

Wielkość w bajtach buforu adresowanego przez pole VCHRP lub VCHRO.

Jeśli struktura MQCHARV jest używana jako pole wyjściowe w wywołaniu funkcji, to pole musi być inicjowane z podaną długością buforu. Jeśli wartość VCHRL jest większa niż VCHRS, do programu wywołującego w buforze zostaną zwrócone tylko bajty danych VCHRS.

Wartość musi być większa lub równa zero lub następująca wartość specjalna, która jest rozpoznawana:

VSUSL
Jeśli określono VSUSL, długość buforu jest pobierana z pola VCHRL w strukturze MQCHARV. Ta wartość specjalna nie jest odpowiednia, jeśli struktura jest używana jako pole wyjściowe i podano bufor. Jest to wartość początkowa tego pola.

Wartości początkowe

Tabela 689. Wartości początkowe MQCHARV dla stałych

Nazwa pola	Nazwa stałej	Wartość stałej
VCHRP	Brak	Wskaźnik pusty lub bajty puste.
VCHRO	Brak	0
VCHRS	VSUSL	-1
VCHRL	Brak	0
VCHRC	CSAPL	-3

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQCHARV Structure
D*
D* Address of variable length string
D VCHRP                   1     16*
D* Offset of variable length string
D VCHRO                  17     20I 0
D* Size of buffer
D VCHRS                  21     24I 0
D* Length of variable length string
D VCHRL                  25     28I 0
D* CCSID of variable length string
D VCHRC                  29     32I 0
```

Ponowna definicja CSAPL

W przeciwieństwie do języków programowania obsługiwanych na innych platformach, język RPG nie ma sposobu na ponowne zdefiniowanie zdefiniowanej stałej, dlatego należy ustawić każdy kod VCHRC specjalnie, jeśli ma być używana wartość inna niż CSAPL.

IBM i MQCIH (nagłówekCICS bridge) w systemie IBM i

Struktura MQCIH opisuje informacje, które mogą być dostępne na początku komunikatu wysyłanego do CICS bridge za pośrednictwem IBM MQ for z/OS.

Przegląd

Nazwa formatu: FMCICS.

Wersja: Bieżąca wersja MQCIH to CIVER2. Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w kolejnych opisach.

Udostępniony plik COPY zawiera najnowszą wersję MQCIH z wartością początkową pola *CIVER* ustawioną na CIVER2.

Zestaw znaków i kodowanie: specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQCIH i danych komunikatu aplikacji:

- Aplikacje nawiązujące połączenie z menedżerem kolejek, do którego należy kolejka CICS bridge , muszą udostępniać strukturę MQCIH, która jest w zestawie znaków i kodowaniu menedżera kolejek. Jest to spowodowane tym, że w tym przypadku nie jest wykonywana konwersja danych struktury MQCIH.
- Aplikacje łączące się z innymi menedżerami kolejek mogą udostępnić strukturę MQCIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań. Konwersja MQCIH jest wykonywana przez odbierającego agenta kanału komunikatów połączonego z menedżerem kolejek, który jest właścicielem kolejki CICS bridge .

  Uwaga: Istnieje jeden wyjątek. Jeśli menedżer kolejek, do którego należy kolejka CICS bridge , używa produktu CICS do kolejkowania rozproszonego, MQCIH musi być w zestawie znaków i kodowaniu menedżera kolejek, do którego należy kolejka CICS bridge .
- Dane komunikatu aplikacji po strukturze MQCIH muszą być w tym samym zestawie znaków i kodowaniu, co struktura MQCIH. Pola *CICSI* i *CIENC* w strukturze MQCIH nie mogą być używane do określania zestawu znaków i kodowania danych komunikatu aplikacji.

  Jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek, użytkownik musi podać wyjście konwersji danych, aby dokonać konwersji danych komunikatu aplikacji.

Użycie: Jeśli wartości wymagane przez aplikację są takie same jak wartości początkowe przedstawione na rysunku Tabela 691 na stronie 1072, a most działa z opcją AUTH=LOCAL lub AUTH=IDENTIFY, strukturę MQCIH można pominąć w komunikacie. We wszystkich innych przypadkach struktura musi być obecna.

Most akceptuje strukturę MQCIH version-1 lub version-2 , ale w przypadku transakcji 3270 należy użyć struktury version-2 .

Aplikacja musi zapewnić, że pola udokumentowane jako pola "żądania" mają odpowiednie wartości w komunikacie wysyłanym do mostu. Te pola są danymi wejściowymi mostu.

Pola udokumentowane jako pola "odpowiedzi" są ustawiane przez CICS bridge w komunikacie odpowiedzi wysyłanym przez most do aplikacji. Informacje o błędach są zwracane w polach *CIRET*, *CIFNC*, *CICC*, *CIREA*i *CIAC* , ale nie wszystkie z nich są ustawiane we wszystkich przypadkach. Tabela 690 na stronie 1063 pokazuje, które pola są ustawione dla różnych wartości *CIRET*.

Tabela 690. Zawartość pól informacji o błędach w strukturze MQCIH

CIRET	***CIFNC***	***CICC***	***CIREA***	***CIAC***
CRC000	-	-	-	-
CRC003	-	-	BBC*	-
CRC002 CRC008	IBM MQ call name (nazwa połączenia)	IBM MQ *CMPCOD*	IBM MQ *REASON*	-
CRC001 CRC006 CRC007 CRC009	CICS EIBFN,	CICS EIBRESP,	CICS EIBRESP2	-
CRC004 CRC005	-	-	-	CICS KOD ABCODE

- “Pola” na stronie 1063
- “Wartości początkowe” na stronie 1072
- “Deklaracja RPG” na stronie 1073

Pola

Struktura MQCIH zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

CIAC (4-bajtowy łańcuch znaków)

Kod nieprawidłowego zakończenia.

Wartość zwracana w tym polu jest istotna tylko wtedy, gdy pole *CIRET* ma wartość CRC005 lub CRC004. Jeśli tak, *CIAC* zawiera wartość CICS ABCODE.

To jest pole odpowiedzi. Długość tego pola jest określona przez LNABNC. Wartością początkową tego pola są 4 znaki odstępu.

Jest to indykator określający, czy deskryptory ADS powinny być wysyłane w żądaniach SEND i RECEIVE BMS. Zdefiniowane są następujące wartości:

ADNONE.
Nie wysyłaj ani nie odbieraj deskryptora ADS.

ADSEND
Wyślij deskryptor ADS.

ADRECV
Odbierz deskryptor ADS.

ADMSGF
Użyj formatu komunikatu dla deskryptora ADS.

Powoduje to wysłanie lub odebranie deskryptora ADS za pomocą długiej postaci deskryptora ADS. Długi formularz zawiera pola, które są wyrównane do 4-bajtowych granic.

Pole *CIADS* należy ustawić w następujący sposób:

- Jeśli deskryptory ADS nie są używane, ustaw pole na ADNONE.

- Jeśli deskryptory ADS *są* używane i mają *taki sam* identyfikator CCSID w każdym środowisku, ustaw w tym polu sumę wartości ADSEND i ADRECV.
- Jeśli deskryptory ADS *są* używane, ale mają *różne* identyfikatory CCSID w każdym środowisku, ustaw w tym polu sumę wartości ADSEND, ADRECV i ADMSGF.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest ADNONE.

CIADS (10-cyfrowa liczba całkowita ze znakiem)

Deskryptor ADS wysyłania/odbierania.

Jest to indykator określający, czy deskryptory ADS powinny być wysyłane w żądaniach SEND i RECEIVE BMS. Zdefiniowane są następujące wartości:

ADNONE.
Nie wysyłaj ani nie odbieraj deskryptora ADS.

ADSEND
Wyślij deskryptor ADS.

ADRECV
Odbierz deskryptor ADS.

ADMSGF
Użyj formatu komunikatu dla deskryptora ADS.

Powoduje to wysłanie lub odebranie deskryptora ADS za pomocą długiej postaci deskryptora ADS. Długi formularz zawiera pola, które są wyrównane do 4-bajtowych granic.

Pole *CIADS* należy ustawić w następujący sposób:

- Jeśli deskryptory ADS nie są używane, ustaw pole na ADNONE.
- Jeśli deskryptory ADS *są* używane i mają *taki sam* identyfikator CCSID w każdym środowisku, ustaw w tym polu sumę wartości ADSEND i ADRECV.
- Jeśli deskryptory ADS *są* używane, ale mają *różne* identyfikatory CCSID w każdym środowisku, ustaw w tym polu sumę wartości ADSEND, ADRECV i ADMSGF.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest ADNONE.

CIAI (4-bajtowy łańcuch znaków)

Klawisz AID.

Jest to początkowa wartość klucza AID w momencie rozpoczęcia transakcji. Jest to wartość jednobajtowa wyrównana do lewej.

Jest to pole żądania używane tylko dla transakcji 3270. Długość tego pola jest określona przez LNATID. Wartością początkową tego pola są 4 odstępy.

CIAUT (8-bajtowy łańcuch znaków)

Hasło lub przepustka.

Jest to hasło lub przepustka. Jeśli uwierzytelnianie za pomocą identyfikatora użytkownika jest aktywne dla CICS bridge, do uwierzytelniania nadawcy komunikatu używany jest identyfikator *CIAUT* z identyfikatorem użytkownika w kontekście tożsamości MQMD.

To jest pole żądania. Długość tego pola jest określona przez LNAUTH. Wartością początkową tego pola jest 8 odstępów.

CICC (10-cyfrowa liczba całkowita ze znakiem)

IBM MQ kod zakończenia lub CICS EIBRESP.

Wartość zwracana w tym polu jest zależna od *CIRET* ; zawiera sekcja Tabela 690 na stronie 1063.

To jest pole odpowiedzi. Wartością początkową tego pola jest CCOK.

CICNC (4-bajtowy łańcuch znaków)

Kod transakcji nieprawidłowego zakończenia.

Jest to kod nieprawidłowego zakończenia, który ma być używany do zakończenia transakcji (zazwyczaj jest to transakcja konwersacyjna żądająca większej ilości danych). W przeciwnym razie pole to jest puste.

Jest to pole żądania używane tylko dla transakcji 3270. Długość tego pola jest określona przez LNCNCL. Wartością początkową tego pola są 4 odstępy.

CICP (10-cyfrowa liczba całkowita ze znakiem)

Pozycja kursora.

Jest to początkowa pozycja kursora w momencie rozpoczęcia transakcji. Później, w przypadku transakcji konwersacyjnych, pozycja kursora znajduje się w wektorze RECEIVE.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest wyświetlane, jeśli wartość *CIVER* jest mniejsza niż CIVER2.

CICSI (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

CICT (10-cyfrowa liczba całkowita ze znakiem)

Określa, czy zadanie może być konwersacyjne.

Jest to indykator określający, czy zadanie powinno mieć możliwość wysyłania żądań dodatkowych informacji, czy też powinno zostać zakończone awaryjnie. Wartość musi być jedną z następujących wartości:

CTYY
Zadanie jest konwersacyjne.

NNRK
Zadanie nie jest konwersacyjne.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest CTNO.

CIENC (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

CIEO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie błędu w komunikacie.

Jest to pozycja niepoprawnych danych wykrytych przez wyjście mostu. W tym polu znajduje się przesunięcie od początku komunikatu do miejsca, w którym znajdują się niepoprawne dane.

Jest to pole odpowiedzi używane tylko dla transakcji 3270. Wartością początkową tego pola jest 0. To pole nie jest wyświetlane, jeśli wartość *CIVER* jest mniejsza niż CIVER2.

CIFAC (8-bajtowy łańcuch bitowy)

Token narzędzia mostu.

Jest to 8-bajtowy token narzędzia mostu. Celem znacznika narzędzia mostu jest umożliwienie wielu transakcjom w pseudokonwersacji użycia tego samego narzędzia mostu (wirtualnego terminalu 3270). W pierwszym komunikacie lub tylko w pierwszym komunikacie w pseudokonwersacji należy ustawić wartość FCNONE; spowoduje to, że program CICS przydzieli nowe narzędzie mostu dla tego komunikatu. Znacznik narzędzia mostu jest zwracany w komunikatach odpowiedzi, jeśli w komunikacie wejściowym określono niezerową wartość *CIFKT* . Kolejne komunikaty wejściowe mogą następnie używać tego samego znacznika narzędzia mostu.

Zdefiniowana jest następująca wartość specjalna:

FCNONE (brak)
Nie określono tokenu BVT.

Jest to zarówno pole żądania, jak i pole odpowiedzi używane tylko dla transakcji 3270. Długość tego pola jest określona przez LNFAC. Wartością początkową tego pola jest FCNONE.

CIFKT (10-cyfrowa liczba całkowita ze znakiem)

Czas zwolnienia obiektu mostu.

Jest to czas w sekundach, przez który narzędzie mostu będzie przechowywane po zakończeniu transakcji użytkownika. W przypadku transakcji niekonwersacyjnych wartość powinna wynosić zero.

Jest to pole żądania używane tylko dla transakcji 3270. Wartością początkową tego pola jest 0.

CIFL (4-bajtowy łańcuch znaków)

Emulowane atrybuty terminalu.

Jest to nazwa zainstalowanego terminalu, który ma być używany jako model dla narzędzia mostu. Wartość pusta oznacza, że wartość *CIFL* jest pobierana z definicji profilu transakcji mostu lub używana jest wartość domyślna.

Jest to pole żądania używane tylko dla transakcji 3270. Długość tego pola jest określona przez LNFACL. Wartością początkową tego pola są 4 odstępy.

CIFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Wartość musi być następująca:

CIFNON
Brak flag.

To jest pole żądania. Wartością początkową tego pola jest CIFNON.

CIFMT (8-bajtowy łańcuch znaków)

Nazwa formatu IBM MQ danych następujących po MQCIH.

Określa nazwę formatu IBM MQ dla danych, które są zgodne ze strukturą MQCIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Ta nazwa formatu jest również używana dla komunikatu odpowiedzi, jeśli pole *CIRFM* ma wartość FMNONE.

- W przypadku żądań DPL *CIFMT* musi być nazwą formatu obszaru COMMAREA.
- W przypadku żądań 3270 *CIFMT* musi mieć wartość CSQCBDCI, a *CIRFM* musi mieć wartość CSQCBDC0.

Wyjścia konwersji danych dla tych formatów muszą być zainstalowane w menedżerze kolejek, w którym mają być uruchamiane.

Jeśli komunikat żądania powoduje wygenerowanie komunikatu odpowiedzi na błąd, komunikat odpowiedzi na błąd ma nazwę formatu FMSTR.

To jest pole żądania. Długość tego pola jest określona przez LNFMT. Wartością początkową tego pola jest FMNONE.

CIFNC (4-bajtowy łańcuch znaków)

Nazwa wywołania IBM MQ lub funkcja CICS EIBFN.

Wartość zwracana w tym polu jest zależna od *CIRET* ; zawiera sekcja Tabela 690 na stronie 1063. Jeśli plik *CIFNC* zawiera nazwę wywołania IBM MQ , możliwe są następujące wartości:

CFCONN,
Wywołanie MQCONN.

CFRGET
Wywołanie MQGET.

KFINQ
Wywołanie MQINQ.

CFOPEN,
Wywołanie MQOPEN.

CFPUT (wprowadzanie)
Wywołanie MQPUT.

CFPUT1
Wywołanie MQPUT1 .

BRAK CFNONE
Brak połączenia.

To jest pole odpowiedzi. Długość tego pola jest określona przez LNFUNC. Wartością początkową tego pola jest CFNONE.

CIGWI (10-cyfrowa liczba całkowita ze znakiem)

Przedział czasu oczekiwania na wywołanie MQGET wydane przez zadanie mostu.

To pole ma zastosowanie tylko wtedy, gdy *CIUOW* ma wartość CUFRST. Umożliwia ona aplikacji wysyłającej określenie przybliżonego czasu (w milisekundach), przez który wywołania MQGET wydane przez most powinny czekać na drugi i kolejny komunikat żądania dla jednostki pracy uruchomionej przez ten komunikat. Powoduje to przesłonięcie domyślnego odstępu czasu oczekiwania używanego przez most. Można stosować następujące wartości specjalne:

WIDFLT (SZEROKOŚĆ)
Domyślny odstęp czasu oczekiwania.

Powoduje to, że CICS bridge oczekuje przez okres określony podczas uruchamiania mostu.

WIULIM
Nieograniczony czas oczekiwania.

To jest pole żądania. Wartością początkową tego pola jest WIDFLT.

CIII (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 0. To pole nie jest wyświetlane, jeśli wartość *CIVER* jest mniejsza niż CIVER2.

CILEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQCIH.

Wartość musi być jedną z następujących wartości:

CILEN1
Długość struktury nagłówka informacji version-1 CICS .

CILEN2
Długość struktury nagłówka informacji version-2 CICS .

Następująca stała określa długość bieżącej wersji:

CILENC
Długość bieżącej wersji struktury nagłówka informacji CICS .

To jest pole żądania. Wartością początkową tego pola jest CILEN2.

CILT (10-cyfrowa liczba całkowita ze znakiem)

Typ powiązania.

Wskazuje typ obiektu, który ma zostać połączony przez most. Wartość musi być jedną z następujących wartości:

LTPROG
Program DPL.

LTTRAN
Transakcja 3270.

To jest pole żądania. Wartością początkową tego pola jest LTPROG.

CINTI (4-bajtowy łańcuch znaków)

Następna transakcja do przyłączenia.

Jest to nazwa następnej transakcji zwróconej przez transakcję użytkownika (zwykle przez EXEC CICS RETURN TRANSID). Jeśli nie ma następnej transakcji, to pole jest puste.

Jest to pole odpowiedzi używane tylko dla transakcji 3270. Długość tego pola jest określona przez LNTRID. Wartością początkową tego pola są 4 odstępy.

CIODL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych wyjściowych obszaru COMMAREA.

Jest to długość danych użytkownika, które mają zostać zwrócone klientowi w komunikacie odpowiedzi. Ta długość obejmuje 8-bajtową nazwę programu. Długość obszaru COMMAREA przekazanego do programu dowiązanego jest maksymalną wartością tego pola i maksymalną długością danych użytkownika w komunikacie żądania pomniejszoną o 8.

Uwaga: Długość danych użytkownika w komunikacie to długość komunikatu *bez* struktury MQCIH.

Jeśli długość danych użytkownika w komunikacie żądania jest mniejsza niż *CIODL*, używana jest opcja DATALENGTH komendy LINK ; pozwala to na efektywne przesłanie funkcji LINK do innego regionu CICS .

Można użyć następującej wartości specjalnej:

OLINPT
Długość danych wyjściowych jest taka sama jak długość danych wejściowych.

Ta wartość może być potrzebna nawet wtedy, gdy nie jest wymagana żadna odpowiedź, aby zapewnić, że obszar COMMAREA przekazany do programu połączonego ma wystarczającą wielkość.

Jest to pole żądania używane tylko dla programów DPL. Początkowa wartość tego pola OLINPT.

CIREA (10-cyfrowa liczba całkowita ze znakiem)

IBM MQ kod przyczyny lub opinii lub CICS EIBRESP2.

Wartość zwracana w tym polu jest zależna od *CIRET* ; zawiera sekcja Tabela 690 na stronie 1063.

To jest pole odpowiedzi. Wartością początkową tego pola jest RCNONE.

CIRET (10-cyfrowa liczba całkowita ze znakiem)

Kod powrotu z mostu.

Jest to kod powrotu z CICS bridge opisujący wynik przetwarzania wykonywanego przez most. Pola *CIFNC*, *CICC*, *CIREA*i *CIAC* mogą zawierać dodatkowe informacje (patrz sekcja Tabela 690 na stronie 1063). Jest to jedna z następujących wartości:

CRC000
(0, X'000 ') Brak błędu.

CRC001
(1, X'001 ') Instrukcja EXEC CICS wykryła błąd.

CRC002
(2, X'002 ') Wywołanie funkcji IBM MQ wykryło błąd.

CRC003
(3, X'003 ') CICS bridge wykrył błąd.

CRC004
(4, X'004 ') CICS bridge zakończyło się nieprawidłowo.

CRC005
(5, X'005 ') Aplikacja została zakończona nieprawidłowo.

CRC006
(6, X'006 ') Wystąpił błąd ochrony.

CRC007
(7, X'007 ') Program niedostępny.

CRC008
(8, X'008 ') Drugi lub późniejszy komunikat w bieżącej jednostce pracy nie został odebrany w określonym czasie.

CRC009
(9, X'009 ') Transakcja niedostępna.

To jest pole odpowiedzi. Wartością początkową tego pola jest CRC000.

CIRFM (8-bajtowy łańcuch znaków)

Nazwa formatu IBM MQ komunikatu odpowiedzi.

Jest to nazwa formatu IBM MQ komunikatu odpowiedzi, który zostanie wysłany w odpowiedzi na bieżący komunikat. Reguły kodowania są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Jest to pole żądania używane tylko dla programów DPL. Długość tego pola jest określona przez LNFMT. Wartością początkową tego pola jest FMNONE.

CIRSI (4-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi wynosić 4 odstępy. Długość tego pola jest określona przez LNRSID.

CIRS1 (8-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 8 odstępów.

CIRS2 (8-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 8 odstępów.

CIRS3 (8-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 8 odstępów.

CIRS4 (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi być równa 0. To pole nie jest wyświetlane, jeśli wartość *CIVER* jest mniejsza niż CIVER2.

CIRTI (4-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość musi wynosić 4 odstępy. Długość tego pola jest określona przez LNTRID.

CISC (4-bajtowy łańcuch znaków)

Kod początkowy transakcji.

Jest to indykator określający, czy most emuluje transakcję terminalu, czy transakcję START. Wartość musi być jedną z następujących wartości:

SSTRT
Uruchom.

SCDATA
Dane początkowe.

SCTERM
Zakończ wprowadzanie danych.

SCNONE.
Brak.

W odpowiedzi z mostu to pole jest ustawione na kod początkowy odpowiedni dla następnego identyfikatora transakcji zawartego w polu *CINTI* . W odpowiedzi możliwe są następujące kody początkowe:

- SSTRT
- SCDATA
- SCTERM

Dla CICS Transaction Server 1.2 to pole jest tylko polem żądania; jego wartość w odpowiedzi jest niezdefiniowana.

W przypadku produktu CICS Transaction Server 1.3 i kolejnych wersji jest to zarówno pole żądania, jak i pole odpowiedzi.

To pole jest używane tylko dla transakcji 3270. Długość tego pola jest określona przez LNSTCO. Wartością początkową tego pola jest SCNONE.

CISID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

CISIDV
Identyfikator struktury nagłówka informacji CICS .

To jest pole żądania. Wartością początkową tego pola jest CISIDV.

CITES (10-cyfrowa liczba całkowita ze znakiem)

Status na końcu zadania.

W tym polu wyświetlany jest status transakcji użytkownika na końcu zadania. Zwracana jest jedna z następujących wartości:

TENOSY
Niezsynchronizowane.

Transakcja użytkownika nie została jeszcze zakończona i nie została zsynchronizowana. W tym przypadku pole *MDMT* w strukturze MQMD to MTRQST.

SKRZYDŁO
Zatwierdź jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona, ale została zsynchronizowana z pierwszą jednostką pracy. W tym przypadku pole *MDMT* w strukturze MQMD ma wartość MTDGRM.

TEBACK
Wycofuje jednostkę pracy.

Transakcja użytkownika nie została jeszcze zakończona. Bieżąca jednostka pracy zostanie wycofana. W tym przypadku pole *MDMT* w strukturze MQMD ma wartość MTDGRM.

TEENDT
Zakończ zadanie.

Transakcja użytkownika została zakończona (lub zakończona awaryjnie). W tym przypadku pole *MDMT* w strukturze MQMD ma wartość MTRPLY.

Jest to pole odpowiedzi używane tylko dla transakcji 3270. Wartością początkową tego pola jest TENOSY.

CITI (4-bajtowy łańcuch znaków)

Transakcja do przyłączenia.

Jeśli *CILT* ma wartość LTTRAN, *CITI* jest identyfikatorem transakcji użytkownika, która ma zostać uruchomiona; w tym przypadku należy podać wartość niepustą.

Jeśli *CILT* ma wartość LTPROG, *CITI* jest kodem transakcji, w ramach którego mają być uruchamiane wszystkie programy w jednostce pracy. Jeśli podana wartość jest pusta, używany jest domyślny kod transakcji (CKBP) mostu CICS DPL. Jeśli wartość nie jest pusta, musi być zdefiniowana w zmiennej CICS jako lokalna transakcja z programem początkowym CSQCBP00. To pole ma zastosowanie tylko wtedy, gdy *CIUOW* ma wartość CUFRST lub CUONLY.

To jest pole żądania. Długość tego pola jest określona przez LNTRID. Wartością początkową tego pola są 4 odstępy.

CIUOW (10-cyfrowa liczba całkowita ze znakiem)

Kontrola jednostki pracy.

Steruje to przetwarzaniem jednostki pracy wykonywanym przez CICS bridge. Można zażądać od mostu uruchomienia pojedynczej transakcji lub jednego lub większej liczby programów w ramach jednostki pracy. Pole wskazuje, czy CICS bridge powinien uruchomić jednostkę pracy, wykonać żądaną funkcję w bieżącej jednostce pracy, czy zakończyć jednostkę pracy, zatwierdzając ją lub cofając. W celu zoptymalizowania przepływów transmisji danych obsługiwane są różne kombinacje.

Wartość musi być jedną z następujących wartości:

TYLKO_KOSTKI
Uruchom jednostkę pracy, wykonaj funkcję, a następnie zatwierdź jednostkę pracy (DPL i 3270).

CUCONT,
Dodatkowe dane dla bieżącej jednostki pracy (tylko 3270).

CUFRST,
Uruchom jednostkę pracy i wykonaj funkcję (tylko DPL).

KUMIDL
Wykonaj funkcję w bieżącej jednostce pracy (tylko DPL).

CULAST
Wykonaj funkcję, a następnie zatwierdź jednostkę pracy (tylko DPL).

CUCMIT
Zatwierdź jednostkę pracy (tylko DPL).

KOPOWSTKI
Wycofuje jednostkę pracy (tylko DPL).

To jest pole żądania. Wartością początkową tego pola jest CUONLY.

CIVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

CIVER1
Version-1 CICS struktura nagłówka informacji.

CIVER2
Version-2 CICS struktura nagłówka informacji.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

CIVERC
Bieżąca wersja struktury nagłówka informacji CICS .

To jest pole żądania. Wartością początkową tego pola jest CIVER2.

Wartości początkowe

Tabela 691. Pola w MQCIH

Nazwa pola	Nazwa stałej	Wartość stałej
CISID	CISIDV	'CIH¬'
CIVER	CIVER2	2
CILEN	CILEN2	180
CIENC	Brak	0
CICSI	Brak	0
CIFMT	BRAK FMNONE	Puste
CIFLG	CIFNON	0
CIRET	CRC000	0
CICC	CKOK	0
CIREA	BRAK RCNONE	0
CIUOW	TYLKO_KOSTKI	273
CIGWI	WIDFLT (SZEROKOŚĆ)	-2
CILT	LTPROG	1
CIODL	OLINPT	-1
CIFKT	Brak	0
CIADS	ADNONE.	0
CICT	NNRK	0
CITES	TENOSY	0
CIFAC	FCNONE (brak)	Wartości null
CIFNC	BRAK CFNONE	Puste
CIAC	Brak	Puste
CIAUT	Brak	Puste
CIRS1	Brak	Puste
CIRFM	BRAK FMNONE	Puste
CIRSI	Brak	Puste
CIRTI	Brak	Puste

Tabela 691. Pola w MQCIH (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
CITI	Brak	Puste
CIFL	Brak	Puste
CIAI	Brak	Puste
CISC	SCNONE.	Puste
CICNC	Brak	Puste
CINTI	Brak	Puste
CIRS2	Brak	Puste
CIRS3	Brak	Puste
CICP	Brak	0
CIEO	Brak	0
CIII	Brak	0
CIRS4	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQCIH Structure
D*
D* Structure identifier
D  CISID                  1      4    INZ('CIH ')
D* Structure version number
D  CIVER                  5      8I 0 INZ(2)
D* Length of MQCIH structure
D  CILEN                  9     12I 0 INZ(180)
D* Reserved
D  CIENC                 13     16I 0 INZ(0)
D* Reserved
D  CICSI                 17     20I 0 INZ(0)
D* MQ format name of data that followsMQCIH
D  CIFMT                 21     28    INZ('        ')
D* Flags
D  CIFLG                 29     32I 0 INZ(0)
D* Return code from bridge
D  CIRET                 33     36I 0 INZ(0)
D* MQ completion code or CICSEIBRESP
D  CICC                  37     40I 0 INZ(0)
D* MQ reason or feedback code, or CICSEIBRESP2
D  CIREA                 41     44I 0 INZ(0)
D* Unit-of-work control
D  CIUOW                 45     48I 0 INZ(273)
D* Wait interval for MQGET call issuedby bridge task
D  CIGWI                 49     52I 0 INZ(-2)
D* Link type
D  CILT                  53     56I 0 INZ(1)
D* Output COMMAREA data length
D  CIODL                 57     60I 0 INZ(-1)
D* Bridge facility release time
D  CIFKT                 61     64I 0 INZ(0)
D* Send/receive ADS descriptor
D  CIADS                 65     68I 0 INZ(0)
D* Whether task can beconversational
D  CICT                  69     72I 0 INZ(0)
D* Status at end of task
D  CITES                 73     76I 0 INZ(0)
D* Bridge facility token
```

```
     D  CIFAC                 77     84    INZ(X'00000000000000-
     D                                     00')
     D* MQ call name or CICS EIBFNfunction
     D  CIFNC                 85     88    INZ('    ')
     D* Abend code
     D  CIAC                  89     92    INZ
     D* Password or passticket
     D  CIAUT                 93    100    INZ
     D* Reserved
     D  CIRS1                101    108    INZ
     D* MQ format name of reply message
     D  CIRFM                109    116    INZ('        ')
     D* Remote CICS system ID to use
     D  CIRSI                117    120    INZ
     D* CICS RTRANSID to use
     D  CIRTI                121    124    INZ
     D* Transaction to attach
     D  CITI                 125    128    INZ
     D* Terminal emulated attributes
     D  CIFL                 129    132    INZ
     D* AID key
     D  CIAI                 133    136    INZ
     D* Transaction start code
     D  CISC                 137    140    INZ('    ')
     D* Abend transaction code
     D  CICNC                141    144    INZ
     D* Next transaction to attach
     D  CINTI                145    148    INZ
     D* Reserved
     D  CIRS2                149    156    INZ
     D* Reserved
     D  CIRS3                157    164    INZ
     D* Cursor position
     D  CICP                 165    168I 0 INZ(0)
     D* Offset of error in message
     D  CIEO                 169    172I 0 INZ(0)
     D* Reserved
     D  CIII                 173    176I 0 INZ(0)
     D* Reserved
     D  CIRS4                177    180I 0 INZ(0)
     D*
```

IBM i MQCMHO (tworzenie opcji uchwytu komunikatu) w systemie IBM i

Struktura **MQCMHO** umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia uchwytów komunikatów.

Przegląd

Przeznaczenie
Struktura jest parametrem wejściowym wywołania **MQCRTMH** .

Zestaw znaków i kodowanie
Dane w systemie **MQCMHO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1074
- "Wartości początkowe" na stronie 1076
- "Deklaracja RPG" na stronie 1076

Pola

Struktura MQCMHO zawiera następujące pola. Pola są opisane w porządku alfabetycznym:

CMOPT (10-cyfrowa liczba całkowita ze znakiem)

Można podać jedną z następujących opcji:

CVAL

Po wywołaniu komendy **MQSETMP** w celu ustawienia właściwości w tym uchwycie komunikatu sprawdzana jest poprawność nazwy właściwości, aby upewnić się, że:

- nie zawiera niepoprawnych znaków.
- nie rozpoczyna się od łańcucha "JMS" lub "usr.JMS", z wyjątkiem następujących przypadków:
  - JMSCorrelationID
  - JMSReplyTo
  - JMSType
  - JMSXGroupID
  - JMSXGroupSeq

  Te nazwy są zarezerwowane dla właściwości JMS .
- nie jest jednym z następujących słów kluczowych, w dowolnej kombinacji wielkich i małych liter:
  - "I"
  - "MIĘDZY"
  - "UCIECZKA"
  - "FAŁSZ"
  - "W"
  - "JEST"
  - "PODOBNE"
  - "NIE"
  - "NULL"
  - "LUB"
  - "PRAWDA"
- nie zaczyna się od "Body." lub "Root." (z wyjątkiem "Root.MQMD.").

Jeśli właściwość ma wartość MQ-defined ("mq. *") i nazwa jest rozpoznawana, pola deskryptora właściwości są ustawione na poprawne wartości dla właściwości. Jeśli właściwość nie zostanie rozpoznana, w polu *Support* deskryptora właściwości zostanie ustawiona wartość **PDSUPO** (więcej informacji na ten temat zawiera sekcja PDSUP).

CMDEFV,

Określa domyślny poziom sprawdzania poprawności nazw właściwości.

Domyślny poziom sprawdzania poprawności jest równoważny poziomowi określonemu przez parametr **CMVAL**.

W przyszłej wersji może zostać zdefiniowana opcja administracyjna, która zmieni poziom sprawdzania poprawności po zdefiniowaniu parametru **CMDEFV** .

Jest to wartość domyślna.

CNOVA

Nie jest sprawdzana poprawność nazwy właściwości. Patrz opis **CMVAL**.

Opcja domyślna: Jeśli żadna z opcji opisanych wcześniej w tej sekcji nie jest wymagana, można użyć następującej opcji:

BRAK CMNONE

Wszystkie opcje przyjmują wartości domyślne. Ta wartość wskazuje, że nie określono żadnych innych opcji. **CMNONE** pomaga w dokumentacji programu; nie jest planowane użycie tej opcji z żadną inną, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **CMDEFV**.

CMSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury; wartość musi być następująca:

CMSIDV
Identyfikator struktury opcji tworzenia uchwytu komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **CMSIDV**.

CMVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

CMVER1
Version-1 tworzenie struktury opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej:

CMVERC (CMVERC)
Bieżąca wersja struktury opcji tworzenia uchwytu komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **CMVER1**.

Wartości początkowe

Tabela 692. Pola w MQCMHO

Nazwa pola	Nazwa stałej	Wartość stałej
CMSID	CMSIDV	'CMHO'
CMVER	CMVER1	1
CMOPT	CMDEFV,	0

Deklaracja RPG

```
D* MQCMHO Structure
D*
D*
D* Structure identifier
D  CMSID                  1      4    INZ('CMHO')
D*
D* Structure version number
D  CMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQCRTMH
D  CMOPT                  9     12I 0 INZ(0)
```

IBM i MQCNO (opcje połączenia) w systemie IBM i

Struktura MQCNO umożliwia aplikacji określenie opcji dotyczących połączenia z lokalnym menedżerem kolejek.

Przegląd

Cel: Struktura jest parametrem wejścia/wyjścia wywołania MQCONNX.

Wersja: Bieżąca wersja MQCNO to CNVER6. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Udostępniony plik COPY zawiera najnowszą wersję obiektu MQCNO, która jest obsługiwana przez środowisko, ale z wartością początkową pola CNVER ustawioną na wartość CNVER1. Aby użyć pól, które

nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu CNVER numer wersji wymaganej wersji.

Zestaw znaków i kodowanie: dane w MQCNO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek systemu **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT.

- "Pola" na stronie 1077
- "Wartości początkowe" na stronie 1082
- "Deklaracja RPG" na stronie 1083

Pola

Struktura MQCNO zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

CCDTUL (10-cyfrowa liczba całkowita ze znakiem)

CCDTUL to długość łańcucha identyfikowanego przez CCDTUP lub CCDTUO, który zawiera URL identyfikujący położenie tabeli kanału połączenia klienckiego, która ma być używana dla połączenia.

Komendy CCDTUL należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client.

Jest to programowa alternatywa dla ustawiania zmiennych środowiskowych MQCHLLIB i MQCHLTAB .

Jeśli aplikacja nie działa jako klient, komenda CCDTUL jest ignorowana.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER6.

CCDTUO (10-cyfrowa liczba całkowita ze znakiem)

CCDTUO to przesunięcie w bajtach od początku struktury MQCNO do łańcucha zawierającego URL identyfikujący położenie tabeli kanału połączenia klienckiego, która ma być używana dla połączenia. Przesunięcie może być dodatnie lub ujemne.

Komendy CCDTUL należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client.

Ważne: Można użyć tylko jednego z CCDTUP i CCDTUO. Wywołanie kończy się niepowodzeniem z kodem przyczyny RC2600 , jeśli oba pola są niezerowe.

Jest to programowa alternatywa dla ustawiania zmiennych środowiskowych MQCHLLIB i MQCHLTAB .

Jeśli aplikacja nie jest uruchomiona jako klient, komenda CCDTUO jest ignorowana.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER6.

CCDTUP (wskaźnik)

CCDTUP jest opcjonalnym wskaźnikiem do łańcucha zawierającego URL, który identyfikuje położenie tabeli kanału połączenia klienckiego, która ma być używana dla połączenia.

Parametru CCDTUP należy używać tylko wtedy, gdy aplikacja wywołująca wywołanie MQCONNX jest uruchomiona jako IBM MQ MQI client.

Ważne: Można użyć tylko jednego z CCDTUP i CCDTUO. Wywołanie kończy się niepowodzeniem z kodem przyczyny RC2600 , jeśli oba pola są niezerowe.

Jest to programowa alternatywa dla ustawiania zmiennych środowiskowych MQCHLLIB i MQCHLTAB .

Jeśli aplikacja nie działa jako klient, komenda CCDTUP jest ignorowana.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER6.

CNAN (28-bajtowy łańcuch znaków)

Nazwa ustawiona przez aplikację w celu zidentyfikowania połączenia z menedżerem kolejek. Wartością początkową pola są znaki o kodzie zero.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER7.

CNCCO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach struktury definicji kanału MQCD od początku struktury MQCNO.

CNCCP (wskaźnik)

Jest to wskaźnik do struktury definicji kanału MQCD.

CNCONID (24-bajtowy łańcuch znaków)

Unikalny identyfikator połączenia. To pole umożliwia menedżerowi kolejek niezawodne zidentyfikowanie procesu aplikacji przez przypisanie mu unikalnego identyfikatora podczas pierwszego połączenia z menedżerem kolejek.

Aplikacje używają identyfikatora połączenia do celów korelacji podczas wykonywania wywołań PUT i GET. Wszystkie połączenia są przypisywane do identyfikatora przez menedżera kolejek, bez względu na to, w jaki sposób połączenie zostało nawiązane.

Istnieje możliwość użycia identyfikatora połączenia w celu wymuszenia zakończenia długo działającej jednostki pracy. W tym celu należy określić identyfikator połączenia za pomocą komendy PCF 'Stop Connection' lub komendy MQSC STOP CONN. Więcej informacji na temat używania tych komend zawierają strony pokrewne.

Początkowa wartość tego pola to 24 puste bajty.

CNCT (128-bajtowy łańcuch bitowy)

Jest to znacznik, który menedżer kolejek wiąże z zasobami, na które ma wpływ aplikacja podczas tego połączenia.

Znacznik połączenia menedżera kolejek.

Każda aplikacja lub instancja aplikacji musi używać innej wartości znacznika, aby menedżer kolejek mógł poprawnie serializować dostęp do odpowiednich zasobów. Więcej szczegółów zawierają opisy opcji CN* CT*. Znacznik przestaje być poprawny, gdy aplikacja kończy działanie, lub wysyła wywołanie MQDISC.

Jeśli znacznik nie jest wymagany, należy użyć następującej wartości specjalnej:

BRAK CTNONE
Nie określono znacznika połączenia.

Wartością długości pola jest zero binarne.

Jest to pole wejściowe. Długość tego pola jest określona przez LNCTAG. Wartością początkową tego pola jest CTNONE. To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER3.

Podczas nawiązywania połączenia z menedżerem kolejek produktu z/OS należy użyć pola ConnTag .

CNNORES2 (4-bajtowy łańcuch znaków)

Pole zastrzeżone umożliwiające dopełnienie struktury do granicy 64-bitowej. Wartość początkowa pola jest zerem binarnym dla długości pola.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER7.

CNOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje, które sterują działaniem MQCONNX.

Opcje powiązania

Opcje powiązania sterują typem używanego powiązania IBM MQ . Należy określić tylko jedną z następujących opcji:

CNSBND (CNSBND)
Powiązanie standardowe.

Standardowa opcja powiązania powoduje, że aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania, zwykle w oddzielnych procesach. Układ utrzymuje integralność menedżera kolejek, czyli chroni menedżera kolejek przed błędnym programem.

Komendy CNSBND należy używać w sytuacjach, gdy aplikacja nie została w pełni przetestowana lub może być nierzetelna lub niegodna zaufania. CNSBND jest wartością domyślną.

Parametr CNSBND jest definiowany w celu wspomagania dokumentacji programu. Nie należy używać tej opcji z żadną inną opcją sterującą typem użytego powiązania, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Ta opcja jest obsługiwana we wszystkich środowiskach.

CNFBND (CNFBND)
Powiązanie krótkiej ścieżki.

Opcja powiązania krótkiej ścieżki powoduje, że aplikacja i agent lokalnego menedżera kolejek są częścią tej samej jednostki wykonywania. Krótka ścieżka różni się od standardowego powiązania, w którym aplikacja i agent lokalnego menedżera kolejek działają w oddzielnych jednostkach wykonywania.

Opcja CNFBND jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania; przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

Opcja CNFBND może być korzystna w sytuacjach, gdy wiele procesów zużywa więcej zasobów niż ogólny zasób używany przez aplikację.Aplikacja używająca powiązania krótkiej ścieżki jest nazywana *aplikacją zaufaną*.

Przy podejmowaniu decyzji o użyciu powiązania krótkiej ścieżki należy wziąć pod uwagę następujące ważne kwestie:

- **Użycie opcji CNFBND nie zapobiega zmianie lub uszkodzeniu przez aplikację komunikatów i innych obszarów danych należących do menedżera kolejek. Tej opcji należy używać tylko w sytuacjach, w których te problemy zostały w pełni przeanalizowane.**
- Aplikacja nie może używać asynchronicznych sygnałów ani przerwań zegara (takich jak `sigkill`) z wartością CNFBND. Istnieją również ograniczenia dotyczące używania segmentów pamięci współużytkowanej.
- Aplikacja nie może mieć jednocześnie więcej niż jednego wątku połączonego z menedżerem kolejek.
- Aplikacja musi użyć wywołania MQDISC , aby rozłączyć się z menedżerem kolejek.
- Aplikacja musi zakończyć działanie przed zakończeniem działania menedżera kolejek za pomocą komendy endmqm .

Poniższe punkty dotyczą użycia CNFBND we wskazanych środowiskach:

- W systemie IBM izadanie musi być uruchomione w profilu użytkownika QMQM , który należy do grupy QMQMADM . Ponadto program nie może zostać zakończony nieprawidłowo, w przeciwnym razie mogą wystąpić nieprzewidywalne rezultaty.

Więcej informacji na temat wpływu używania zaufanych aplikacji zawiera sekcja Nawiązywanie połączenia z menedżerem kolejek przy użyciu wywołania MQCONNX oraz sekcja Ograniczenia dotyczące zaufanych aplikacji.

CNSHBD (CNSHBD)
Powiązania współużytkowane.

Opcja powiązań współużytkowanych powoduje, że aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania, zwykle w oddzielnych procesach. Układ utrzymuje integralność menedżera kolejek, czyli chroni menedżera kolejek przed błędnym programem. Jednak niektóre zasoby są współużytkowane przez aplikację

i agenta lokalnego menedżera kolejek. Parametr CNSHBD jest ignorowany, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

CNIBND
Izolowane powiązania.

Opcja izolowanych powiązań powoduje, że aplikacja i agent lokalnego menedżera kolejek są uruchamiane w oddzielnych jednostkach wykonywania, zwykle w oddzielnych procesach. Układ utrzymuje integralność menedżera kolejek, czyli chroni menedżera kolejek przed błędnym programem. Proces aplikacji i agent lokalnego menedżera kolejek są od siebie odizolowane, ponieważ nie współużytkują zasobów. Opcja CNIBND jest ignorowana, jeśli menedżer kolejek nie obsługuje tego typu powiązania. Przetwarzanie jest kontynuowane tak, jakby opcja nie została określona.

Opcje współużytkowania uchwytu

Poniższe opcje sterują współużytkowaniem uchwytów między różnymi wątkami (jednostkami przetwarzania równoległego) w ramach tego samego procesu. Można podać tylko jedną z tych opcji.

CNHSN (CNHSN)
Brak obsługi współużytkowania między wątkami.

Opcja współużytkowania braku uchwytu między wątkami wskazuje, że uchwyty połączeń i obiektów mogą być używane tylko przez wątek, który spowodował przydzielenie uchwytu; jest to wątek, który wywołał wywołanie MQCONN, MQCONNXlub MQOPEN . Uchwyty nie mogą być używane przez inne wątki należące do tego samego procesu.

CNHSB (CNHSB)
Współużytkowanie uchwytu szeregowego między wątkami z blokowaniem wywołań.

Opcja współużytkowania uchwytu szeregowego między wątkami z blokowaniem wywołań wskazuje, że uchwyty połączenia i obiektu przydzielone przez jeden wątek procesu mogą być używane przez inne wątki należące do tego samego procesu. Jednak tylko jeden wątek na raz może używać konkretnego uchwytu, czyli dozwolone jest tylko użycie uchwytu szeregowego. Jeśli wątek próbuje użyć uchwytu, który jest już używany przez inny wątek, wywołanie blokuje (oczekuje) aż uchwyt stanie się dostępny.

CNHSNB (CNHSNB)
Współużytkowanie uchwytu szeregowego między wątkami bez blokowania wywołań.

Opcja współużytkowania uchwytu szeregowego między wątkami, bez blokowania wywołań, jest taka sama jak opcja " *z opcją* blocking ", z tą różnicą, że jeśli uchwyt jest używany przez inny wątek, wywołanie natychmiast kończy się z opcją CCFAIL i RC2219 zamiast blokowania do czasu, aż uchwyt stanie się dostępny.

Wątek może mieć zero lub jeden niewspółużytkowany uchwyt oraz zero lub więcej współużytkowanych uchwytów:

- Każde wywołanie MQCONN lub MQCONNX , które określa CNHSN , zwraca nowy niewspółużytkowany uchwyt w pierwszym wywołaniu i ten sam niewspółużytkowany uchwyt w kolejnych wywołaniach (przy założeniu, że nie jest to wywołanie MQDISC). Kod przyczyny dla drugiego i późniejszych wywołań to RC2002 .
- Każde wywołanie MQCONNX , które określa parametr CNHSB lub CNHSNB , zwraca nowy uchwyt współużytkowany dla każdego wywołania.

Uchwyty obiektów dziedziczą te same właściwości współużytkowania, co uchwyt połączenia określony w wywołaniu MQOPEN , który utworzył uchwyt obiektu. Ponadto jednostki pracy dziedziczą te same właściwości współużytkowania, co uchwyt połączenia używany do uruchamiania jednostki pracy. Jeśli jednostka pracy jest uruchamiana w jednym wątku przy użyciu uchwytu współużytkowanego, jednostka pracy może być aktualizowana w innym wątku przy użyciu tego samego uchwytu.

Jeśli opcja współużytkowania uchwytu nie zostanie określona, środowisko będzie określać wartość domyślną:

- W środowisku Microsoft Transaction Server (MTS) wartość domyślna jest taka sama jak wartość CNHSB.
- W innych środowiskach wartość domyślna jest taka sama jak CNHSN.

Opcje ponownego połączenia

Opcje ponownego połączenia określają, czy połączenie można ponownie nawiązać. Można ponownie nawiązywać połączenia tylko z klientami.

CNRCDF (CNRCDF)
Opcja ponownego połączenia jest tłumaczona na wartość domyślną. Jeśli nie ustawiono wartości domyślnej, wartość tej opcji jest ustawiana na DISABLED. Wartość tej opcji jest przekazywana do serwera i może być odpytywana przez **PCF** i **MQSC**.

CNRC (kod CNRC)
Aplikację można ponownie połączyć z dowolnym menedżerem kolejek zgodnie z wartością parametru MQCONNX **QMNAME** . Opcji CNRC należy używać tylko wtedy, gdy nie ma powinowactwa między aplikacją kliencką a menedżerem kolejek, z którym początkowo nawiązywane jest połączenie. Wartość tej opcji jest przekazywana do serwera i może być odpytywana przez **PCF** i **MQSC**.

CNRCD (CNRCD)
Nie można ponownie połączyć aplikacji. Wartość opcji nie jest przekazywana do serwera.

CNRCQM (CNRCQM)
Aplikację można ponownie połączyć tylko z menedżerem kolejek, z którym pierwotnie była połączona. Tej wartości należy użyć, jeśli można ponownie nawiązać połączenie z klientem, ale istnieje powinowactwo między aplikacją kliencką a menedżerem kolejek, z którym pierwotnie nawiązał połączenie. Wartość tę należy wybrać, jeśli klient ma automatycznie nawiązywać ponowne połączenie z instancją rezerwową menedżera kolejek o wysokiej dostępności. Wartość tej opcji jest przekazywana do serwera i może być odpytywana przez **PCF** i **MQSC**.

Opcji CNRC, CNRCDi CNRCQM należy używać tylko dla połączeń klienckich. Jeśli opcje są używane dla połączenia powiązania, operacja MQCONNX kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_OPTIONS_ERROR.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest wymagana, można użyć następującej opcji:

BRAK
Nie określono żadnych opcji.

Wartość CNNONE jest definiowana w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją CN* , ale ponieważ jej wartość wynosi zero, nie można jej wykryć.

CNSCO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach struktury MQSCO od początku struktury MQCNO.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER4.

CNSCP (wskaźnik)

Jest to adres struktury MQSCO.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER4.

CNSECPO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie parametrów zabezpieczeń. Przesunięcie struktury MQCSP używane do określania identyfikatora użytkownika i hasła.

Wartość może być dodatnia lub ujemna. Wartością początkową tego pola jest 0.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER5.

CNSECPP (wskaźnik)

Wskaźnik parametrów ochrony. Adres struktury MQCSP używanej do określania identyfikatora użytkownika i hasła.

Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

To pole jest ignorowane, jeśli wartość CNVER jest mniejsza niż CNVER5.

CNSID (4-bajtowy łańcuch znaków)

Identyfikator struktury struktury MQCNO.

Wartość musi być następująca:

CNSIDV
Identyfikator struktury opcji połączenia.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CNSIDV.

CNVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury dla struktury MQCNO.

Wartość musi być następująca:

CNVER6
Version-6 -struktura opcji połączenia.

Ta wersja jest obsługiwana we wszystkich środowiskach.

CNVER7
Struktura opcji połączenia Version-7 .

Ta wersja jest obsługiwana we wszystkich środowiskach.

Następująca stała określa numer wersji bieżącej:

CNVERC
Bieżąca wersja struktury connect-options.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CNVER7.

Wartości początkowe

Tabela 693. Pola w MQCNO

Nazwa pola	Nazwa stałej	Wartość stałej
CNSID	CNSIDV	'CNO¬
CNVER	CNVER5	1
CNOPT	BRAK	0
CNCCO	Brak	0
CNCCP	Brak	Pusty wskaźnik lub puste bajty
CNCT	BRAK CTNONE	Wartości null
CNSCP	Brak	Pusty wskaźnik lub puste bajty
CNSCO	Brak	0
CNCONID	Brak	Wartości null
CNSECPO	Brak	0
CNSECPP	Brak	Pusty wskaźnik lub puste bajty

Tabela 693. Pola w MQCNO (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
CCDTUL	Brak	0
CCDTUO	Brak	0
CCDTUP	Brak	Pusty wskaźnik lub puste bajty

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*****************************************************************
     D**                                                            **
     D**                   IBM MQ for IBM i                         **
     D**                                                            **
     D**  FILE NAME:      CMQCNOG                                   **
     D**                                                            **
     D**  DESCRIPTION:    MQCNO Structure -- Connect Options        **
     D**                                                            **
     D*****************************************************************
     D**  <N_OCO_COPYRIGHT>                                         **
     D**  Licensed Materials - Property of IBM                      **
     D**                                                            **
     D**  5724-H72                                                  **
     D**  (c) Copyright IBM Corp. 1993, 2024. All Rights Reserved.  **
     D**                                                            **
     D**  US Government Users Restricted Rights - Use, duplication or **
     D**  disclosure restricted by GSA ADP Schedule Contract with   **
     D**  IBM Corp.                                                 **
     D**  <NOC_COPYRIGHT>                                           **
     D*****************************************************************
     D**                                                            **
     D**  FUNCTION:       This file declares the structure MQCNO,   **
     D**                  which is used by the main MQI.            **
     D**                                                            **
     D**  PROCESSOR:      RPG (ILE)                                 **
     D**                                                            **
     D*****************************************************************
     D*
     D*
     D*****************************************************************
     D** <BEGIN_BUILDINFO>                                          **
     D** Generated on:  08/02/16 13:50                             **
     D** Build Level:   L000000                                    **
     D** Build Type:    Production                                 **
     D** Pointer Size:  128 Bit                                    **
     D** Source File:                                              **
     D** CMQCNOG                                                   **
     D** <END_BUILDINFO>                                           **
     D*****************************************************************
     D*
     D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     D*
     D*
     D* MQCNO Structure
     D*
     D* Structure identifier
     D  CNSID                  1      4    INZ('CNO ')
     D* Structure version number
     D  CNVER                  5      8I 0 INZ(1)
     D* Options that control the action of MQCONNX
     D  CNOPT                  9     12I 0 INZ(0)
     D* Ver:1 **
     D* Offset of MQCD structure for client connection
     D  CNCCO                 13     16I 0 INZ(0)
     D* Address of MQCD structure for client connection
     D  CNCCP                 17     32*   INZ(*NULL)
     D* Ver:2 **
     D* Queue managerconnection tag
     D  CNCT                  33    160    INZ(X'00000000000000000-
     D                                     000000000000000000000000-
     D                                     000000000000000000000000-
```

```
D                                      00000000000000000000000000-
D                                      00000000000000000000000000-
D                                      00000000000000000000000000-
D                                      00000000000000000000000000-
D                                      00000000000000000000000000-
D                                      00000000000000000000000000-
D                                      00000000000000000000000000-
D                                      00000000000')
D* Ver:3 **
D* Address of MQSCO structure for client connection
D  CNSCP                161    176*   INZ(*NULL)
D* Offset of MQSCO structure for client connection
D  CNSCO                177    180I 0 INZ(0)
D* Ver:4 **
D* Unique Connection Identifier
D  CNCONID              181    204    INZ(X'00000000000000000-
D                                      00000000000000000000000000-
D                                      000000')
D* Offset of MQCSP structure
D  CNSECPO              205    208I 0 INZ(0)
D* Address of MQCSP structure
D  CNSECPP              209    224*   INZ(*NULL)
D* Ver:5 **
D* Address of CCDT URL string
D  CNCCDTUP             225    240*   INZ(*NULL)
D* Offset of CCDT URL string
D  CNCCDTUO             241    244I 0 INZ(0)
D* Length of CCDT URL
D  CNCCDTUL             245    248I 0 INZ(0)
D* Ver:6 **
D*
D*************************************************************
D**  End of CMQCNOG                                         **
D*************************************************************
```

IBM i MQCSP (parametry zabezpieczeń) w systemie IBM i

Podsumowanie struktury MQCSP dla produktu IBM i.

Przegląd

Cel: Struktura MQCSP umożliwia usłudze autoryzacji uwierzytelnienie identyfikatora użytkownika i hasła. Strukturę parametrów zabezpieczeń połączenia MQCSP określa się w wywołaniu MQCONNX.

Zestaw znaków i kodowanie: dane w MQCSP muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu **CodedCharSetId** oraz kodowanie lokalnego menedżera kolejek określone przez ENNAT.

- "Pola" na stronie 1084
- "Wartości początkowe" na stronie 1086
- "Deklaracja RPG" na stronie 1087

Pola

Struktura MQCSP zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

CSAUTHT (10-cyfrowa liczba całkowita ze znakiem)

Jest to typ wykonywanego uwierzytelniania.

Poprawne wartości:

Sieć CSAN
Nie używaj pól ID użytkownika i hasła.

CAUIAP
Pola identyfikatora i hasła użytkownika są uwierzytelniane.

Jest to pole wejściowe. Wartością początkową tego pola jest CSAN.

CSCPPL (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość hasła, które ma być używane w uwierzytelnianiu.

Maksymalna długość hasła nie zależy od platformy. Jeśli długość hasła jest większa niż dozwolona, żądanie uwierzytelniania kończy się niepowodzeniem z kodem powrotu RC2035.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

CSCPPO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach hasła, które ma być używane w uwierzytelnianiu.

Przesunięcie może być dodatnie lub ujemne.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

CSCPPP (wskaźnik)

Jest to adres hasła, które ma być używane podczas uwierzytelniania.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty.

CSCSPUIL (10-cyfrowa liczba całkowita ze znakiem)

Jest to długość identyfikatora użytkownika, który ma być używany podczas uwierzytelniania.

Maksymalna długość identyfikatora użytkownika nie zależy od platformy. Jeśli długość identyfikatora użytkownika jest większa niż dozwolona, żądanie uwierzytelniania kończy się niepowodzeniem z kodem powrotu RC2035.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

CSCSPUIO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie (w bajtach) identyfikatora użytkownika, który ma być używany w uwierzytelnianiu.

Przesunięcie może być dodatnie lub ujemne.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

CSCSPUIP (wskaźnik)

Jest to adres ID użytkownika, który ma być używany podczas uwierzytelniania.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość CSVER jest mniejsza niż CSVER5.

V 9.3.0 V 9.3.0 **CSINITKL (10-cyfrowa liczba całkowita ze znakiem)**

Jest to długość klucza początkowego dla systemu zabezpieczenia hasłem.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

V 9.3.0 V 9.3.0 **CSINITKO (10-cyfrowa liczba całkowita ze znakiem)**

Jest to przesunięcie w bajtach klucza początkowego dla systemu zabezpieczenia hasłem. Przesunięcie może być dodatnie lub ujemne.

Do określenia klucza początkowego można użyć wartości *CSINITKO* lub *CSINITKP* , ale nie obu tych wartości jednocześnie. Więcej informacji na ten temat zawiera opis pola *CSINITKP* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

V 9.3.0 V 9.3.0 **CSINITKP (wskaźnik)**

Jest to adres (w bajtach) początkowego klucza systemu ochrony hasła.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty.

IBM MQ MQI clients może podać wartości niektórych pól jako wartości, które zostały zaszyfrowane za pomocą systemu ochrony hasłem IBM MQ . Następujące pola mogą zawierać zaszyfrowane wartości:

- Hasło repozytorium kluczy w strukturze MQSCO.

Klucz początkowy jest używany przez algorytm szyfrowania do szyfrowania i deszyfrowania tych wartości. Jeśli klucz początkowy zostanie podany, gdy wartości tych pól są szyfrowane za pomocą programu narzędziowego **runmqicred** , klient musi określić ten sam klucz początkowy podczas nawiązywania połączenia z menedżerem kolejek.

Klucz początkowy określony za pomocą tego pola nadpisuje klucz początkowy określony za pomocą zmiennej środowiskowej *MQS_MQI_KEYFILE* lub właściwości *MQIInitialKeyFile* w sekcji Security pliku konfiguracyjnego klienta.

Do określenia klucza początkowego można użyć wartości *CSINITKO* lub *CSINITKP* , ale nie obu tych wartości jednocześnie.

CSRE1 (4-bajtowy łańcuch znaków)

Pole zastrzeżone, wymagane do wyrównania wskaźnika w systemie IBM i.

Jest to pole wejściowe. Początkowa wartość tego pola jest pusta.

CSRS2 (8-bajtowy łańcuch znaków)

Pole zastrzeżone, wymagane do wyrównania wskaźnika w systemie IBM i.

Jest to pole wejściowe. Początkowa wartość tego pola jest pusta.

CSSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

CSSIDV
Identyfikator struktury parametrów zabezpieczeń.

CSVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

CSVER1
Struktura parametrów zabezpieczeń Version-1 .

V 9.3.0 V 9.3.0 V 9.3.0 **CSVER2**
Struktura parametrów zabezpieczeń Version-2 .

Następująca stała określa numer wersji bieżącej:

CVERC
Bieżąca wersja struktury parametrów zabezpieczeń.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CSVER1.

Wartości początkowe

Tabela 694. Pola w MQCNO

Nazwa pola	Nazwa stałej	Wartość stałej
CSSID	CSSIDV	'CSP¬'
CSVER	CSVER1	1
CSAUTHT	Brak	0
CSRE1	Brak	Wartości null

Tabela 694. Pola w MQCNO (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
CSCSPUIP	Brak	Wskaźnik pusty
CSCSPUIO	Brak	0
CSCSPUIL	Brak	0
CSRS2	Brak	Wartości null
CSCPPP	Brak	Wskaźnik pusty
CSCPPO	Brak	0
CSCPPL	Brak	0
V 9.3.0 V 9.3.0 *CSINITKP*	Brak	Wskaźnik pusty
V 9.3.0 V 9.3.0 *CSINITKO*	Brak	0
V 9.3.0 V 9.3.0 *CSINITKL*	Brak	0

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
      D*
      D* MQCSP Structure
      D*
      D* Structure identifier
      D  CSSID                  1      4    INZ('CSP ')
      D* Structure version number
      D  CSVER                  5      8I 0 INZ(1)
      D* Type of authentication
      D  CSAUTHT                9     12I 0 INZ(0)
      D* Reserved
      D  CSRE1                 13     16    INZ(X'00000000')
      D* Address of user ID
      D  CSCSPUIP              17     32*   INZ(*NULL)
      D* Offset of user ID
      D  CSCSPUIO              33     36I 0 INZ(0)
      D* Length of user ID
      D  CSCSPUIL              37     40I 0 INZ(0)
      D* Reserved
      D  CSRS2                 41     48    INZ(X'0000000000000000')
      D* Address of password
      D  CSCPPP                49     64*   INZ(*NULL)
      D* Offset of password
      D  CSCPPO                65     68I 0 INZ(0)
      D* Length of password
      D  CSCPPL                69     72I 0 INZ(0)
```

IBM i MQCTLO (struktura opcji wywołania zwrotnego sterowania) w systemie IBM i

Struktura określająca funkcję zwrotną elementu sterującego.

Przegląd

Przeznaczenie
Struktura MQCTLO służy do określania opcji związanych z funkcją zwrotną sterowania.

Struktura jest parametrem wejściowym i wyjściowym wywołania MQCTL .

Wersja
Bieżąca wersja MQCTLO to CTLV1.

Zestaw znaków i kodowanie
Dane w obiekcie MQCTLO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

- "Pola" na stronie 1088
- "Wartości początkowe" na stronie 1089
- "Deklaracja RPG" na stronie 1089

Pola

Struktura MQCTLO zawiera następujące pola; pola są opisane w porządku alfabetycznym:

COCONNAREA (10-cyfrowa liczba całkowita ze znakiem)

Struktura opcji sterujących-pole ConnectionArea .

Jest to pole dostępne do użycia przez funkcję zwrotną.

Menedżer kolejek nie podejmuje żadnych decyzji na podstawie zawartości tego pola i jest przekazywany bez zmian z pola CBCCONNAREA w strukturze MQCBC, która jest parametrem wywołania MQCB.

To pole jest ignorowane dla wszystkich operacji innych niż CTLSR i CTLSW.

Jest to pole wejściowe i wyjściowe funkcji zwrotnej. Wartością początkową tego pola jest wskaźnik pusty lub bajty puste.

COOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje sterujące działaniem MQCTLO.

CTLFQ,
Wymuś niepowodzenie wywołania MQCTLO, jeśli menedżer kolejek lub połączenie jest w stanie wyciszania.

Określ wartość GMFIQ w opcjach MQGMO przekazanych w wywołaniu MQCB, aby spowodować powiadomienie konsumentów komunikatów podczas wyciszania.

TLTHR
Ta opcja informuje system, że aplikacja wymaga, aby wszystkie konsumenty komunikatów dla tego samego połączenia były wywoływane w tym samym wątku.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

CTLNO
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne. Parametr CTLNO jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest planowane, aby ta opcja była używana z innymi opcjami, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Jest to pole wejściowe. Wartością początkową pola *COOPT* jest CTLNO.

CORSV (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest znakiem odstępu.

COSID (10-cyfrowa liczba całkowita ze znakiem)

Struktura opcji sterujących-pole StrucId .

Jest to identyfikator struktury; wartość musi być następująca:

CTLSI
Identyfikator struktury opcji sterujących.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CTLSI.

COVER (10-cyfrowa liczba całkowita ze znakiem)

Struktura opcji sterujących-pole Wersja.

Jest to numer wersji struktury; wartość musi być następująca:

CTLV1
Version-1 -struktura opcji sterujących.

Następująca stała określa numer wersji bieżącej:

CTLCV
Bieżąca wersja struktury opcji sterujących.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest CTLV1.

Wartości początkowe

Tabela 695. Pola w MQCTLO

Nazwa pola	Nazwa stałej	Wartość stałej
COSID	CTLSI	'CTLO'
COVER	CTLV1	1
COOPT	CTLNO	Wartości null
CORSV	Pole zastrzeżone	
COCONNAREA	Brak	Pusty wskaźnik lub puste bajty

Deklaracja RPG

```
D* MQCTLO Structure
D*
D*
D* Structure identifier
D  COSID                  1      4    INZ('CTLO')
D*
D* Structure version number
D  COVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQCTL
D  COOPT                  9     12I 0 INZ(0)
D*
D* Reserved
D  CORSV                 13     16I 0 INZ(-1)
D*
D* MQCTL Data area passed to the function
D  COCONNAREA            17     32*   INZ(*NULL)
```

IBM i MQDH (nagłówek dystrybucji) w systemie IBM i

Struktura MQDH opisuje dodatkowe dane, które są obecne w komunikacie, gdy komunikat jest komunikatem listy dystrybucyjnej przechowywanym w kolejce transmisji.

Przegląd

Cel: Komunikat listy dystrybucyjnej to komunikat wysyłany do wielu kolejek docelowych. Dodatkowe dane składają się ze struktury MQDH, po której następuje tablica rekordów MQOR i tablica rekordów MQPMR.

Ta struktura jest używana przez wyspecjalizowane aplikacje, które umieszczają komunikaty bezpośrednio w kolejkach transmisji lub usuwają komunikaty z kolejek transmisji (na przykład agenty kanału komunikatów).

Ta struktura nie powinna być używana przez zwykłe aplikacje, które po prostu chcą umieszczać komunikaty na listach dystrybucyjnych. Aplikacje te powinny używać struktury MQOD do definiowania miejsc docelowych na liście dystrybucyjnej oraz struktury MQPMO do określania właściwości komunikatów lub do odbierania informacji o komunikatach wysyłanych do poszczególnych miejsc docelowych.

Zestaw znaków i kodowanie: Dane w MQDH muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQDH muszą zostać ustawione w polach *MDCSI* i *MDENC* w następujących polach:

- MQMD (jeśli struktura MQDH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQDH (wszystkie inne przypadki).

Użycie: Jeśli aplikacja umieszcza komunikat na liście dystrybucyjnej, a niektóre lub wszystkie miejsca docelowe są zdalne, menedżer kolejek dodaje do danych komunikatu aplikacji przedrostek w postaci struktur MQXQH i MQDH oraz umieszcza komunikat w odpowiedniej kolejce transmisji. Dlatego dane występują w następującej kolejności, gdy komunikat znajduje się w kolejce transmisji:

- Struktura MQXQH
- Struktura MQDH plus tablice rekordów MQOR i MQPMR
- Dane komunikatu aplikacji

W zależności od miejsc docelowych menedżer kolejek może wygenerować więcej niż jeden taki komunikat i umieścić go w różnych kolejkach transmisji. W tym przypadku struktury MQDH w tych komunikatach identyfikują różne podzbiory miejsc docelowych zdefiniowanych przez listę dystrybucyjną otwartą przez aplikację.

Aplikacja, która umieszcza komunikat listy dystrybucyjnej bezpośrednio w kolejce transmisji, musi być zgodna z wcześniej opisaną sekwencją i musi mieć pewność, że struktura MQDH jest poprawna. Jeśli struktura MQDH nie jest poprawna, menedżer kolejek może wybrać niepowodzenie wywołania MQPUT lub MQPUT1 z kodem przyczyny RC2135.

Komunikaty mogą być przechowywane w kolejce w formie listy dystrybucyjnej tylko wtedy, gdy kolejka jest zdefiniowana jako zdolna do obsługi komunikatów listy dystrybucyjnej (patrz atrybut kolejki **DistLists** opisany w sekcji “Atrybuty kolejek” na stronie 1410). Jeśli aplikacja umieszcza komunikat listy dystrybucyjnej bezpośrednio w kolejce, która nie obsługuje list dystrybucyjnych, menedżer kolejek dzieli komunikat listy dystrybucyjnej na pojedyncze komunikaty i umieszcza je w kolejce.

- “Pola” na stronie 1090
- “Wartości początkowe” na stronie 1093
- “Deklaracja RPG” na stronie 1094

Pola

Struktura MQDH zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

DHCNT (10-cyfrowa liczba całkowita ze znakiem)

Liczba rekordów MQOR.

Definiuje liczbę miejsc docelowych. Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *DHCNT* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

DHCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych następujących po rekordach MQOR i MQPMR.

Określa identyfikator zestawu znaków danych następujących po tablicach rekordów MQOR i MQPMR. Nie ma on zastosowania do danych znakowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola *MDPAT* w MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

DHENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe danych następujących po rekordach MQOR i MQPMR.

Określa kodowanie liczbowe danych następujących po tablicach rekordów MQOR i MQPMR. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQDH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

DHFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi ogólne.

Można podać następującą opcję:

DHFNOWY
Wygeneruj nowe identyfikatory komunikatów.

Ta opcja wskazuje, że dla każdego miejsca docelowego na liście dystrybucyjnej ma zostać wygenerowany nowy identyfikator komunikatu. Tę opcję można ustawić tylko wtedy, gdy nie ma rekordów put-message lub gdy rekordy są obecne, ale nie zawierają pola *PRMID* .

Użycie tej flagi powoduje odraczanie generowania identyfikatorów komunikatów do ostatniego możliwego momentu, a mianowicie momentu, w którym komunikat listy dystrybucyjnej jest ostatecznie dzielony na pojedyncze komunikaty. Minimalizuje to ilość informacji sterujących, które muszą być przesyłane z komunikatem listy dystrybucyjnej.

Gdy aplikacja umieszcza komunikat na liście dystrybucyjnej, menedżer kolejek ustawia DHFNEW w MQDH, która jest generowana, gdy spełnione są oba poniższe warunki:

- Aplikacja nie udostępnia rekordów komunikatów wstawianych lub podane rekordy nie zawierają pola *PRMID* .
- Pole *MDMID* w strukturze MQMD ma wartość MINONE lub pole *PMOPT* w strukturze MQPMO zawiera identyfikator PMNMID.

Jeśli nie są potrzebne żadne opcje, można podać następujące informacje:

DHFNON
Brak flag.

Ta stała wskazuje, że nie określono żadnych flag. DHFNON jest zdefiniowane w celu wspomagania dokumentacji programu. Ta stała nie jest przeznaczona do użycia z żadną inną, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Wartością początkową tego pola jest DHFNON.

DHFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych następujących po rekordach MQOR i MQPMR.

Określa nazwę formatu danych, które są zgodne z tablicami rekordów MQOD i MQPMR (w zależności od tego, co nastąpi później).

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

DHLEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQDH plus następujące rekordy MQOR i MQPMR.

Jest to liczba bajtów od początku struktury MQDH do początku danych komunikatu po tablicach rekordów MQOR i MQPMR. Dane występują w następującej kolejności:

- Struktura MQDH
- Tablica rekordów MQOR
- Tablica rekordów MQPMR
- Dane komunikatu

Tablice rekordów MQOR i MQPMR są adresowane za pomocą przesunięć zawartych w strukturze MQDH. Jeśli te przesunięcia powodują powstanie nieużywanych bajtów między jedną lub większą liczbą struktur MQDH, tablicami rekordów i danymi komunikatu, nieużywane bajty muszą zostać uwzględnione w wartości *DHLEN*, ale treść tych bajtów nie jest zachowywana przez menedżer kolejek. Tablica rekordów MQPMR może występować przed tablicą rekordów MQOR.

Wartością początkową tego pola jest 0.

DHORO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu MQOR od początku MQDH.

To pole określa przesunięcie w bajtach pierwszego rekordu w tablicy rekordów obiektów MQOR zawierających nazwy kolejek docelowych. W tej tablicy znajdują się rekordy *DHCNT* . Te rekordy (plus wszystkie bajty pominięte między pierwszym rekordem obiektu a poprzednim polem) są uwzględniane w długości określonej przez pole *DHLEN* .

Lista dystrybucyjna musi zawsze zawierać co najmniej jedno miejsce docelowe, dlatego wartość *DHORO* musi być zawsze większa od zera.

Wartością początkową tego pola jest 0.

DHPRF (10-cyfrowa liczba całkowita ze znakiem)

Flagi wskazujące, które pola MQPMR są obecne.

Można podać co najmniej jedną z następujących opcji:

Identyfikator PFMID
Pole identyfikatora komunikatu jest obecne.

Identyfikator PFCID
Pole identyfikatora korelacji jest obecne.

Identyfikator PFGID
Pole identyfikatora grupy jest obecne.

PFFB
Pole informacji zwrotnej jest obecne.

FACC
Rozliczanie-pole tokenu jest obecne.

Jeśli nie ma pól MQPMR, można podać następujące informacje:

PPFNONE
Brak pól rekordu komunikatu umieszczania (put-message).

PFNONE jest zdefiniowane w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta stała była używana z żadną inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Wartością początkową tego pola jest PFNONE.

DHPRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu MQPMR od początku MQDH.

W tym polu jest wyświetlane przesunięcie w bajtach pierwszego rekordu w tablicy rekordów komunikatów umieszczania MQPMR zawierających właściwości komunikatu. Jeśli istnieje, w tej tablicy znajdują się rekordy *DHCNT* . Te rekordy (plus wszystkie bajty pominięte między pierwszym rekordem komunikatu wstawianego a poprzednim polem) są uwzględniane w długości określonej przez pole *DHLEN* .

Rekordy umieszczania komunikatów są opcjonalne. Jeśli nie podano żadnych rekordów, parametr *DHPRO* ma wartość zero, a parametr *DHPRF* ma wartość PFNONE.

Wartością początkową tego pola jest 0.

DHSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

DHSIDV
Identyfikator struktury nagłówka dystrybucji.

Wartością początkową tego pola jest DHSIDV.

DHVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

DHVER1
Numer wersji struktury nagłówka dystrybucji.

Następująca stała określa numer wersji bieżącej:

DHVERC,
Bieżąca wersja struktury nagłówka dystrybucji.

Wartością początkową tego pola jest DHVER1.

Wartości początkowe

Tabela 696. Pola w MQDH

Nazwa pola	Nazwa stałej	Wartość stałej
DHSID	DHSIDV	'DH¬¬'

Tabela 696. Pola w MQDH (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
DHVER	DHVER1	1
DHLEN	Brak	0
DHENC	Brak	0
DHCSI	CSUNDF	0
DHFMT	BRAK FMNONE	Puste
DHFLG	DHFNON	0
DHPRF	PPFNONE	0
DHCNT	Brak	0
DHORO	Brak	0
DHPRO	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDH Structure
D*
D* Structure identifier
D  DHSID                  1      4    INZ('DH  ')
D* Structure version number
D  DHVER                  5      8I 0 INZ(1)
D* Length of MQDH structure plusfollowing MQOR and MQPMR records
D  DHLEN                  9     12I 0 INZ(0)
D* Numeric encoding of data that followsthe MQOR and MQPMR records
D  DHENC                 13     16I 0 INZ(0)
D* Character set identifier of data thatfollows the MQOR and MQPMR
D* records
D  DHCSI                 17     20I 0 INZ(0)
D* Format name of data that follows theMQOR and MQPMR records
D  DHFMT                 21     28    INZ('        ')
D* General flags
D  DHFLG                 29     32I 0 INZ(0)
D* Flags indicating which MQPMR fieldsare present
D  DHPRF                 33     36I 0 INZ(0)
D* Number of MQOR records present
D  DHCNT                 37     40I 0 INZ(0)
D* Offset of first MQOR record from startof MQDH
D  DHORO                 41     44I 0 INZ(0)
D* Offset of first MQPMR record fromstart of MQDH
D  DHPRO                 45     48I 0 INZ(0)
```

IBM i MQDLH (nagłówek niedostarczonego komunikatu) w systemie IBM i

Przegląd

Przeznaczenie

Struktura MQDLH opisuje informacje, które są przedrostkiem danych komunikatu aplikacji dla komunikatów w kolejce niedostarczonych komunikatów. Komunikat może pojawić się w kolejce niedostarczonych komunikatów, ponieważ menedżer kolejek lub agent kanału komunikatów przekierował go do kolejki. Aplikacja może umieścić komunikat bezpośrednio w kolejce.

Nazwa formatu
FMDLH

Zestaw znaków i kodowanie
Plik MQDLH może znajdować się na początku danych komunikatu aplikacji. Jeśli tak, pola w strukturze MQDLH są w zestawie znaków i kodowaniu określonym przez pola MDCSI i MDENC . Jeśli nie, zestaw znaków i kodowanie są ustawiane przez pola MDCSI i MDENC w strukturze nagłówka, która poprzedza MQDLH.

Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Użycie
Aplikacje, które umieszczają komunikaty bezpośrednio w kolejce niedostarczonych komunikatów, muszą poprzedzać dane komunikatu strukturą MQDLH i inicjować pola odpowiednimi wartościami. Jednak menedżer kolejek nie wymaga obecności struktury MQDLH lub określenia poprawnych wartości dla pól.

Jeśli komunikat jest zbyt długi, aby umieścić go w kolejce niedostarczonych komunikatów, aplikacja musi rozważyć wykonanie jednej z następujących czynności:

- Obetnij dane komunikatu, aby zmieściły się w kolejce niedostarczonych komunikatów.
- Zapisz komunikat w pamięci dyskowej i umieść komunikat raportu o wyjątku w kolejce niedostarczonych komunikatów, wskazując, że komunikat jest zbyt długi.
- Odrzuć komunikat i zwróć błąd do jego twórcy. Jeśli komunikat jest komunikatem krytycznym. Odrzuć komunikat tylko wtedy, gdy wiadomo, że nadawca nadal ma kopię komunikatu. Na przykład komunikat odebrany przez agenta kanału komunikatów z kanału komunikacyjnego.

Wybór odpowiedniej opcji zależy od projektu aplikacji.

Menedżer kolejek wykonuje przetwarzanie specjalne, gdy komunikat, który jest segmentem, jest umieszczany ze strukturą MQDLH na początku. Więcej informacji na ten temat zawiera opis struktury MQMDE .

- “Umieszczanie komunikatów w kolejce niedostarczonych komunikatów” na stronie 1095
- “Pobieranie komunikatów z kolejki niedostarczonych komunikatów” na stronie 1096
- “Pola” na stronie 1096
- “Wartości początkowe” na stronie 1100
- “Deklaracja RPG” na stronie 1100

Umieszczanie komunikatów w kolejce niedostarczonych komunikatów

Jeśli komunikat jest umieszczany w kolejce niedostarczonych komunikatów, struktura MQMD używana dla wywołania MQPUT lub MQPUT1 musi być taka sama, jak struktura MQMD powiązana z komunikatem. MQMD jest zwykle zwracany przez wywołanie MQGET , z wyjątkiem następujących przypadków:

- Pola MDCSI i MDENC muszą być ustawione na dowolny zestaw znaków i kodowanie używane dla pól w strukturze MQDLH .
- Pole MDFMT musi być ustawione na wartość FMDLH , aby wskazać, że dane rozpoczynają się od struktury MQDLH .
- Pola kontekstu MDACC, MDAID, MDAOD, MDPAN, MDPAT, MDPD, MDPTi MDUID muszą być ustawione przy użyciu opcji kontekstu odpowiedniej dla danych okoliczności:
  - Aplikacja umieszczająca w kolejce niedostarczonych komunikatów komunikat, który nie jest powiązany z żadnym poprzednim komunikatem, musi użyć opcji PMDEFC . Opcja PMDEFC powoduje, że menedżer kolejek ustawia wszystkie pola kontekstu w deskryptorze komunikatu na ich wartości domyślne.
  - Aplikacja serwera, która umieszcza odebrany komunikat w kolejce niedostarczonych komunikatów, musi użyć opcji PMPASA , aby zachować oryginalne informacje o kontekście.

– Aplikacja serwera umieszczając w kolejce niedostarczonych komunikatów odpowiedź na odebrany komunikat musi korzystać z opcji PMPASI . Opcja PMPASI zachowuje informacje o tożsamości, ale ustawia informacje o pochodzeniu na informacje o aplikacji serwera.
– Agent kanału komunikatów umieszczający w kolejce niedostarczonych komunikatów komunikat odebrany z kanału komunikacyjnego musi używać opcji PMSETA . Opcja PMSETA zachowuje oryginalne informacje o kontekście.

W samej strukturze MQDLH pola muszą być ustawione w następujący sposób:

- Pola DLCSI, DLENCi *DLFMT* muszą być ustawione na wartości opisujące dane zgodne ze strukturą MQDLH . Te wartości są zwykle wartościami z oryginalnego deskryptora komunikatu.
- Pola kontekstu DLPAT, DLPAN, DLPDi DLPT muszą być ustawione na wartości odpowiednie dla aplikacji, która umieszcza komunikat w kolejce niedostarczonych komunikatów. Te wartości nie są powiązane z oryginalnym komunikatem.
- Inne pola muszą być odpowiednio ustawione.

Aplikacja musi zapewnić, że wszystkie pola mają poprawne wartości i że pola znakowe są dopełniane spacjami do zdefiniowanej długości pola. Dane znakowe nie mogą być przedwcześnie zakończone znakiem o kodzie zero. Menedżer kolejek nie przekształca wartości NULL i kolejnych znaków w znaki odstępu w strukturze MQDLH .

Pobieranie komunikatów z kolejki niedostarczonych komunikatów

Aplikacje, które pobierają komunikaty z kolejki niedostarczonych komunikatów, muszą sprawdzić, czy komunikaty zaczynają się od struktury MQDLH . Aplikacja może określić, czy struktura MQDLH jest obecna, sprawdzając pole MDFMT w deskryptorze komunikatu MQMD. Jeśli pole ma wartość FMDLH, dane komunikatu rozpoczynają się od struktury MQDLH . Komunikaty w kolejce niedostarczonych komunikatów mogą zostać obcięte, jeśli były pierwotnie zbyt długie dla kolejki, dla której były przeznaczone.

Pola

Struktura MQDLH zawiera następujące pola; pola są opisane w porządku alfabetycznym:

DLCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych następujących po MQDLH.

DLCSI określa identyfikator zestawu znaków dla danych o strukturze MQDLH . Dane pochodzą zwykle z oryginalnego komunikatu. Nie ma zastosowania do danych znakowych w samej strukturze MQDLH .

W wywołaniu funkcji MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CSINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych następujących po tej strukturze znajdują się w tym samym zestawie znaków, co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie występuje błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET .

Parametr CSINHT nie może być używany, jeśli wartością pola MDPAT w pliku MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

DLDM (48-bajtowy łańcuch znaków)

Nazwa oryginalnego docelowego menedżera kolejek.

Jest to nazwa menedżera kolejek, który był oryginalnym miejscem docelowym komunikatu.

Długość tego pola jest określona przez wartość LNQMN. Wartością początkową tego pola jest 48 znaków odstępu.

DLDQ (48-bajtowy łańcuch znaków)

Nazwa oryginalnej kolejki docelowej.

Jest to nazwa kolejki komunikatów, która była oryginalnym miejscem docelowym komunikatu.

Długość tego pola jest określona przez wartość LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

DLENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe danych następujących po MQDLH.

DLENC określa kodowanie liczbowe danych, które są zgodne ze strukturą MQDLH . Dane pochodzą zwykle z oryginalnego komunikatu. Nie ma zastosowania do danych liczbowych w samej strukturze MQDLH .

W wywołaniu funkcji MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

DLFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych następująca po MQDLH.

Określa nazwę formatu danych, które są zgodne ze strukturą MQDLH (zwykle są to dane z oryginalnego komunikatu).

W wywołaniu funkcji MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły dla pola MDFMT w programie MQMD.

Długość tego pola jest określona przez wartość LNFMT. Wartością początkową tego pola jest FMNONE.

DLPAN (28-bajtowy łańcuch znaków)

Nazwa aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów.

Format nazwy zależy od pola DLPAT . Patrz opis pola MDPAN w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.

Jeśli jest to menedżer kolejek, który przekierowuje komunikat do kolejki niedostarczonych komunikatów, nazwa DLPAN zawiera pierwsze 28 znaków nazwy menedżera kolejek. W razie potrzeby nazwa jest uzupełniana odstępami.

Długość tego pola jest określona przez wartość LNPAN. Wartością początkową tego pola jest 28 znaków odstępu.

DLPAT (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów.

To pole ma takie samo znaczenie jak pole MDPAT w deskryptorze komunikatu MQMD (szczegółowe informacje zawiera sekcja “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141).

Jeśli jest to menedżer kolejek, który przekierowuje komunikat do kolejki niedostarczonych komunikatów, parametr DLPAT ma wartość ATQM.

Wartością początkową tego pola jest 0.

DLPD (8-bajtowy łańcuch znaków)

Data umieszczenia komunikatu w kolejce niedostarczonych komunikatów.

Format używany dla daty wygenerowania tego pola przez menedżer kolejek to:

- YYYYMMDD

gdzie znaki reprezentują:

YYYY
rok (cztery cyfry)

MM
miesiąc roku (od 01 do 12)

DD
dzień miesiąca (od 01 do 31)

Czas Greenwich (GMT) jest używany w polach DLPD i DLPT , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na GMT.

Długość tego pola jest określona przez wartość LNPDAT. Wartością początkową tego pola jest osiem pustych znaków.

DLPT (8-bajtowy łańcuch znaków)

Czas umieszczenia komunikatu w kolejce niedostarczonych komunikatów.

Format używany dla czasu wygenerowania tego pola przez menedżer kolejek to:

- HHMMSSTH

gdzie znaki reprezentują (w kolejności):

HH
godzin (od 00 do 23)

MM
minuty (od 00 do 59)

SS
sekundy (od 00 do 59; patrz uwaga w dalszej części tego tematu)

T
dziesiąte części sekundy (od 0 do 9)

H
setne sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z dokładnym standardem czasu, możliwe jest zwrócenie wartości 60 lub 61 dla sekund w polu DLPT. Dodatkowa sekunda występuje, gdy sekundy przestępne są wstawiane do globalnego standardu czasu.

Czas Greenwich (GMT) jest używany w polach DLPD i DLPT , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na GMT.

Długość tego pola jest określona przez wartość LNPTIM. Wartością początkową tego pola jest osiem pustych znaków.

DLREA (10-cyfrowa liczba całkowita ze znakiem)

Przyczyna pojawienia się komunikatu w kolejce niedostarczonych komunikatów.

Identyfikuje to przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów zamiast w oryginalnej kolejce docelowej. Musi to być jedna z wartości FB* lub RC* (na przykład RC2053). Szczegółowe informacje na temat typowych wartości FB*, które mogą wystąpić, zawiera opis pola *MDFB* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141 .

Jeśli wartość należy do zakresu od FBIFST do FBILST, rzeczywisty kod błędu IMS można określić, odejmując wartość FBIERR od wartości pola *DLREA* .

Niektóre wartości FB* występują tylko w tym polu. Odnoszą się one do komunikatów repozytorium, komunikatów wyzwalacza lub komunikatów kolejki transmisji, które są przesyłane do kolejki niedostarczonych komunikatów. Są to następujące wartości:

FBABEG
Nie można uruchomić aplikacji.

Aplikacja przetwarzająca komunikat wyzwalacza nie mogła uruchomić aplikacji o nazwie podanej w polu TMAI komunikatu wyzwalacza; patrz sekcja “MQTM-komunikat wyzwalacza” na stronie 1273.

FBATYP
Błąd typu aplikacji.

Aplikacja przetwarzająca komunikat wyzwalacza nie mogła uruchomić aplikacji, ponieważ pole TMAT komunikatu wyzwalacza jest niepoprawne; patrz “MQTM-komunikat wyzwalacza” na stronie 1273.

FBBOCD
Usunięto kanał odbiorczy klastra.

Komunikat znajdował się w kolejce transmisji klastra przeznaczonej dla kolejki klastra, która została otwarta z opcją FBIERR . Zdalny kanał odbiorczy klastra, który ma być używany do przesyłania komunikatu do kolejki docelowej, został usunięty przed wysłaniem komunikatu. Ponieważ określono FBIERR , do przesłania komunikatu można użyć tylko kanału wybranego podczas otwierania kolejki. Ponieważ ten kanał nie jest już dostępny, komunikat został umieszczony w kolejce niedostarczonych komunikatów.

FBNARM
Komunikat nie jest komunikatem repozytorium.

FBSBCX
Komunikat został zatrzymany przez wyjście automatycznej definicji kanału.

FBSBMX
Komunikat został zatrzymany przez wyjście komunikatu kanału.

FBTM
Struktura MQTM jest niepoprawna lub nie istnieje.

Pole MDFMT w pliku MQMD określa FMTM, ale komunikat nie rozpoczyna się od poprawnej struktury MQTM . Na przykład mnemonik *TMSID* może nie być poprawny. Możliwe, że nie rozpoznano *TMVER* . Długość komunikatu wyzwalacza może być niewystarczająca do umieszczenia struktury MQTM .

FBXQME
Komunikat w kolejce transmisji ma niepoprawny format.

Agent kanału komunikatów wykrył, że komunikat w kolejce transmisji ma niepoprawny format. Agent kanału komunikatów umieszcza komunikat w kolejce niedostarczonych komunikatów przy użyciu tego kodu sprzężenia zwrotnego.

Wartością początkową tego pola jest RCNONE.

DLSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

DLSIDV
Identyfikator struktury nagłówka niedostarczonego komunikatu.

Wartością początkową tego pola jest DLSIDV.

DLVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

DLVER1
Numer wersji dla struktury nagłówka niedostarczonego komunikatu.

Następująca stała określa numer wersji bieżącej:

DLVERC
Bieżąca wersja struktury nagłówka niedostarczonego komunikatu.

Wartością początkową tego pola jest DLVER1.

Wartości początkowe

Tabela 697. Zmienne w MQDLH

Nazwa pola	Nazwa stałej	Wartość stałej
Identyfikator DLSID	DLSIDV	'DLH¬'
SERWER	DLVER1	1
DLREA (DLREA)	RCNONE	0
Kolejka DLDQ	Brak	Puste
DLDM (DLDM)	Brak	Puste
DLENC (DLENC)	Brak	0
DLCSI (protokół DLCSI)	CSUNDF	0
DLFMT (DLFMT)	FMNONE	Puste
DLPAT (DLPAT)	Brak	0
DLPAN (DLPAN)	Brak	Puste
DLPD (DLPD)	Brak	Puste
DLPT (DLPT)	Brak	Puste

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDLH Structure
D*
D* Structure identifier
D  DLSID                  1      4    INZ('DLH ')
D* Structure version number
D  DLVER                  5      8I 0 INZ(1)
D* Reason message arrived on dead-letter(undelivered-message) queue
D  DLREA                  9     12I 0 INZ(0)
D* Name of original destination queue
D  DLDQ                  13     60    INZ
D* Name of original destination queue manager
D  DLDM                  61    108    INZ
D* Numeric encoding of data that followsMQDLH
D  DLENC                 109    112I 0 INZ(0)
D* Character set identifier of data thatfollows MQDLH
D  DLCSI                 113    116I 0 INZ(0)
D* Format name of data that followsMQDLH
D  DLFMT                 117    124    INZ('        ')
D* Type of application that put messageon dead-letter
D* (undelivered-message)queue
D  DLPAT                 125    128I 0 INZ(0)
D* Name of application that put messageon dead-letter
D* (undelivered-message)queue
D  DLPAN                 129    156    INZ
D* Date when message was put ondead-letter (undelivered-message)queue
D  DLPD                  157    164    INZ
D* Time when message was put on thedead-letter (undelivered-message)queue
D  DLPT                  165    172    INZ
```

IBM i MQDMHO (opcje usuwania uchwytu komunikatu) w systemie IBM i

Struktura **MQDMHO** umożliwia aplikacjom określanie opcji sterujących usuwaniem uchwytów komunikatów.

Przegląd

Cel: Struktura jest parametrem wejściowym wywołania **MQDLTMH** .

Zestaw znaków i kodowanie: dane w pliku **MQDMHO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1101
- "Wartości początkowe" na stronie 1101
- "Deklaracja RPG" na stronie 1102

Pola

Struktura MQDMHO zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

DMOPT (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

BRAK DMNONE

Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **DMNONE**.

DMSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury; wartość musi być następująca:

DMSIDV

Identyfikator struktury opcji usuwania uchwytu komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **DMSIDV**.

DMVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

DMVER1
Version-1 usunięcie struktury opcji uchwytu komunikatu.

Następująca stała określa numer wersji bieżącej:

DMVERC,
Bieżąca wersja struktury opcji usuwania uchwytu komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **DMVER1**.

Wartości początkowe

Tabela 698. Pola w MQDMHO

Nazwa pola	Nazwa stałej	Wartość stałej
DMSID	DMSIDV	'DMHO'
DMVER	DMVER1	1
DMOPT	BRAK DMNONE	0

Deklaracja RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID                  1      4    INZ('DMHO')
D*
D* Structure version number
D  DMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT                  9     12I 0 INZ(0)
```

IBM i MQDMPO (usuwanie opcji właściwości komunikatu) w systemie IBM i

Struktura definiująca opcje właściwości usuwania komunikatu.

Przegląd

Cel: Struktura MQDMPO umożliwia aplikacjom określanie opcji sterujących sposobem usuwania właściwości komunikatów. Struktura jest parametrem wejściowym wywołania MQDLTMP.

Zestaw znaków i kodowanie: Dane w MQDMPO muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1102
- "Wartości początkowe" na stronie 1103
- "Deklaracja RPG" na stronie 1103

Pola

Struktura MQDMPO zawiera następujące pola. Pola są opisane w porządku alfabetycznym:

DPOPT (10-cyfrowa liczba całkowita ze znakiem)

Usuń strukturę opcji właściwości komunikatu-pole DPOPT.

Opcje położenia: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości.

DDPDELF,
Usuwa pierwszą właściwość, która jest zgodna z podaną nazwą.

DPDELC,
Powoduje usunięcie właściwości wskazywanej przez kursor właściwości; jest to właściwość, do której ostatnio wysłano zapytanie za pomocą opcji IPINQF lub IPINQN.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany. Jest on również resetowany, gdy uchwyt komunikatu jest określony w polu *HMSG* MQGMO w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany lub gdy uchwyt komunikatu jest określony w polu *HMSG* struktury MQGMO w strukturze MQGET w wywołaniu MQGET lub w strukturze MQPMO w wywołaniu MQPUT.

Wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i przyczyną RC2471 , jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze ustawiony. Działanie to kończy się również niepowodzeniem z tymi kodami, jeśli właściwość wskazywała na kursor właściwości, która została już usunięta.

Jeśli żadna z tych opcji nie jest wymagana, można użyć następującej opcji:

DPNONE
Nie określono żadnych opcji.

Wartością początkową tego pola wejściowego jest DPDELF.

DPSID (10-cyfrowa liczba całkowita ze znakiem)

Usuń strukturę opcji właściwości komunikatu-pole DPSID.

Jest to identyfikator struktury. Wartość musi być następująca:

DPSIDV
Identyfikator struktury opcji usuwania właściwości komunikatu.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest DPSIDV.

DPVER (10-cyfrowa liczba całkowita ze znakiem)

Usuń strukturę opcji właściwości komunikatu-pole DPVER.

Jest to numer wersji struktury. Wartość musi być następująca:

DPVER1
Numer wersji struktury opcji usuwania właściwości komunikatu.

Następująca stała określa numer wersji bieżącej:

DPVERC,
Bieżąca wersja struktury opcji usuwania właściwości komunikatu.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest DPVER1.

Wartości początkowe

Tabela 699. Pola w MQDPMO

Nazwa pola	Nazwa stałej	Wartość stałej
DPSID	DPSIDV	'DMPO'
DPVER	DPVER1	1
DPOPT	Opcje sterujące działaniem komendy MQDLTMP	DPNONE

Deklaracja RPG

```
D* MQDMPO Structure
D*
D*
D* Structure identifier
D  DPSID                  1      4    INZ('DMPO')
D*
D* Structure version number
D  DPVER                  5      8I 0 INZ(1)
D*
** Options that control the action of
D* MQDLTMP
D  DPOPT                  9     12I 0 INZ(0)
```

IBM i MQEPH (wbudowany nagłówek PCF) w systemie IBM i

Przegląd

Przeznaczenie
Struktura MQEPH opisuje dodatkowe dane, które są obecne w komunikacie, gdy komunikat jest komunikatem w formacie programowalnej komendy (PCF). Pole *EPPFH* definiuje parametry PCF, które są zgodne z tą strukturą i umożliwia śledzenie danych komunikatu PCF z innymi nagłówkami.

Nazwa formatu
EPFMT

Zestaw znaków i kodowanie
Dane w programie MQEPH muszą być w zestawie znaków i kodowaniu lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek systemu **CCSID** .

Ustaw zestaw znaków i kodowanie MQEPH w polach *MDCSI* i *MDENC* w następujących polach:

- MQMD (jeśli struktura MQEPH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQEPH (wszystkie inne przypadki).

Użycie
Struktur MQEPH nie można używać do wysyłania komend do serwera komend lub innego serwera akceptującego PCF menedżera kolejek.

Podobnie serwer komend lub dowolny inny serwer akceptujący PCF menedżera kolejek nie generuje odpowiedzi ani zdarzeń zawierających struktury MQEPH.

- "Pola" na stronie 1104
- "Wartości początkowe" na stronie 1106
- "Deklaracja RPG" na stronie 1106

Pola

Struktura MQEPH zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

EPCSI (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator zestawu znaków danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF. Nie ma on zastosowania do danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest EPCUND.

EPENC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kodowanie liczbowe danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF. Nie ma ono zastosowania do danych znakowych w samej strukturze MQEPH.

Wartością początkową tego pola jest 0.

EPFLG (10-cyfrowa liczba całkowita ze znakiem)

Dozwolone są następujące wartości:

BRAK
Nie określono żadnych opcji. *MDCSI* EPNONE jest zdefiniowane w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta stała była używana z żadną inną, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

EPCSEM,
Zestaw znaków parametrów zawierających dane znakowe jest określany indywidualnie w polu *CCSID* w każdej strukturze. Zestaw znaków pól *EPSID* i *EPFMT* jest definiowany przez parametr *CCSID* w strukturze nagłówka, która poprzedza strukturę MQEPH, lub przez pole *MDCSI* w strukturze MQMD, jeśli MQEPH jest na początku komunikatu.

Wartością początkową tego pola jest EPNONE.

EPFMT (8-bajtowy łańcuch znaków)

Jest to nazwa formatu danych, które są zgodne ze strukturą MQEPH i powiązanymi parametrami PCF.

Wartością początkową tego pola jest EPFMNO.

EPLEN (10-cyfrowa liczba całkowita ze znakiem)

Jest to ilość danych poprzedzających następną strukturę nagłówka. i zapewnia Klientowi:

- Długość nagłówka MQEPH
- Długość wszystkich parametrów PCF następujących po nagłówku
- Puste dopełnianie po tych parametrach

EPLEN musi być wielokrotnością 4.

Część struktury o stałej długości jest zdefiniowana przez EPSTLF.

Wartością początkową tego pola jest 68.

EPPCFH (MQCFH)

Jest to nagłówek formatu komend programowalnych (PCF), definiujący parametry PCF zgodne ze strukturą MQEPH. Umożliwia to śledzenie danych komunikatu PCF z innymi nagłówkami.

Nagłówek PCF jest początkowo zdefiniowany z następującymi wartościami:

Tabela 700. Pola w EPPCFH

Nazwa pola	Nazwa stałej	Wartość stałej
EP3TYP	CFTNON	0
EP3LEN	ŚR.	36
EP3VER	FHVER3	3
EP3CMD	BRAK CMNONE	0
EP3SEQ	Brak	1
EP3CTL	CFCLST	1
EEP3CC	CKOK	0
EP3REA	BRAK RCNONE	0
EP3CNT	Brak	0

Aplikacja musi zmienić typ EP3TYP z CFTNON na poprawny typ struktury na potrzeby używania osadzonego nagłówka PCF.

EPSID (4-bajtowy łańcuch znaków)

Wartość musi być następująca:

Identyfikator EPSTID
Identyfikator struktury nagłówka osadzonego PCF.

Wartością początkową tego pola jest EPSTID.

EPVER (10-cyfrowa liczba całkowita ze znakiem)

Możliwe wartości:

EPVER1
Numer wersji dla osadzonej struktury nagłówka PCF.

Następująca stała określa numer wersji bieżącej:

EPVER3
Bieżąca wersja wbudowanej struktury nagłówka PCF.

Wartością początkową tego pola jest EPVER3.

Wartości początkowe

Tabela 701. Pola w MQEPH		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
EPSID	Identyfikator EPSTID	'EP¬¬'
EPVER	EPVER1	1
EPLEN	EPSTLF,	68
EPENC	Brak	0
EPCSI	EPCUND	0
EPFMT	Nr EPFMNO	Puste
EPFLG	BRAK	0
EPPCFH	Nazwy i wartości zdefiniowane w pliku Tabela 700 na stronie 1105	0

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQEPH Structure
D*
D* Structure identifier
D  EPSID                  1      4
D* Structure version number
D  EPVER                  5      8I 0
D* Total lenght of MQEPH including MQCFHand parameter structures
D* that follow
D  EPLEN                  9     12I 0
D* Numeric encoding of data that follows last PCF parameter structure
D  EPENC                 13     16I 0
D* Character set identifier of data that follows last PCF parameter
D* structure
D  EPCSI                 17     20I 0
D* Format name of data that follows last PCF parameter structure
D  EPFMT                 21     28
D* Flags
D  EPFLG                 29     32I 0
D* Programmable Command Format Header
D  EP3TYP                33     36I 0
D  EP3LEN                37     40I 0
D  EP3VER                41     44I 0
D  EP3CMD                45     48I 0
D  EP3SEQ                49     52I 0
D  EP3CTL                53     56I 0
D  EP3CC                 57     60I 0
D  EP3REA                61     64I 0
D  EP3CNT                65     68I 0
```

IBM i MQGMO (opcje pobierania komunikatów) w systemie IBM i

Struktura MQGMO umożliwia aplikacji określenie opcji sterujących usuwaniem komunikatów z kolejek.

Przegląd

Przeznaczenie

Struktura jest parametrem wejścia/wyjścia w wywołaniu MQGET.

Wersja
Bieżąca wersja produktu MQGMO to GMVER4. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Udostępniony plik COPY zawiera najnowszą wersję produktu MQGMO obsługiwaną przez środowisko, ale z wartością początkową pola *GMVER* ustawioną na wartość GMVER1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *GMVER* numer wersji wymaganej wersji.

Zestaw znaków i kodowanie
Dane w MQGMO muszą być w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

- "Pola" na stronie 1107
- "Wartości początkowe" na stronie 1128
- "Deklaracja RPG" na stronie 1128

Pola

Struktura MQGMO zawiera następujące pola; pola są opisane w porządku alfabetycznym:

GMGST (łańcuch znaków o długości 1 bajtu)

Flaga wskazująca, czy pobrany komunikat należy do grupy.

Ma jedną z następujących wartości:

GSNIG@ item: indo
Komunikat nie należy do grupy.

GSMIG
Komunikat znajduje się w grupie, ale nie jest ostatnim w grupie.

GSLMIG
Komunikat jest ostatnim w grupie.

Ta wartość jest również zwracana, jeśli grupa składa się tylko z jednego komunikatu.

To pole jest polem wyjściowym. Wartością początkową tego pola jest GSNIG. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER2.

GMMH (10-cyfrowa liczba całkowita ze znakiem)

uchwyt komunikatu

Jeśli określono opcję GMPRAQ, a atrybut kolejki PRPCTL nie jest ustawiony na PRPRFH, to jest to uchwyt komunikatu, który jest zapełniany właściwościami komunikatu pobieranego z kolejki. Uchwyt jest tworzony przez wywołanie MQCRTMH. Wszystkie właściwości, które są już powiązane z uchwytem, są czyszczone przed pobraniem komunikatu.

Można również podać następującą wartość:
MQHM_NONE
Nie podano uchwytu komunikatu.

Deskryptor komunikatu nie jest wymagany w wywołaniu MQGET, jeśli podano poprawny uchwyt komunikatu i został on użyty w danych wyjściowych w celu umieszczenia właściwości komunikatu. Deskryptor komunikatu powiązany z uchwytem komunikatu jest używany dla pól wejściowych.

Jeśli w wywołaniu MQGET określono deskryptor komunikatu, ma on zawsze pierwszeństwo przed deskryptorem komunikatu powiązanym z uchwytem komunikatu.

Jeśli określono wartość GMPRRF lub określono wartość GMPRAQ, a atrybut kolejki PRPCTL ma wartość PRPRFH, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2026 , jeśli nie określono parametru deskryptora komunikatu.

Po powrocie z wywołania MQGET właściwości i deskryptor komunikatu powiązane z tym uchwytem komunikatu są aktualizowane w celu odzwierciedlenia stanu pobranego komunikatu (a także deskryptora komunikatu, jeśli został on podany w wywołaniu MQGET). Właściwości komunikatu można następnie uzyskać za pomocą wywołania MQINQMP.

Z wyjątkiem rozszerzeń deskryptora komunikatu, jeśli istnieje, właściwość, do której można uzyskać dostęp za pomocą wywołania MQINQMP, nie jest zawarta w danych komunikatu. Jeśli komunikat w kolejce zawiera właściwości w danych komunikatu, są one usuwane z danych komunikatu przed zwróceniem danych do aplikacji.

Jeśli nie podano uchwytu komunikatu lub wersja jest niższa niż GMVER4 , należy podać poprawny deskryptor komunikatu w wywołaniu MQGET. Wszystkie właściwości komunikatu (z wyjątkiem tych, które są zawarte w deskryptorze komunikatu) są zwracane w temacie danych komunikatu do wartości opcji właściwości w strukturze MQGMO i w atrybucie kolejki PRPCTL.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest HMNONE. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER4.

GMMO (10-cyfrowa liczba całkowita ze znakiem)

Opcje sterujące kryteriami wyboru używanymi dla wywołania MQGET.

Te opcje umożliwiają aplikacji wybór pól parametru **MSGDSC** używanych do wybierania komunikatu zwracanego przez wywołanie MQGET. Aplikacja ustawia wymagane opcje w tym polu, a następnie ustawia odpowiednie pola w parametrze **MSGDSC** na wartości wymagane dla tych pól. Tylko komunikaty, które mają te wartości w deskryptorze MQMD dla komunikatu, są kandydatami do pobrania przy użyciu tego parametru **MSGDSC** w wywołaniu MQGET. Pola, dla których nie określono odpowiedniej opcji dopasowania, są ignorowane podczas wybierania komunikatu, który ma zostać zwrócony. Jeśli w wywołaniu MQGET nie mają być używane żadne kryteria wyboru (oznacza to, że każdy komunikat jest akceptowalny), parametr *GMMO* powinien być ustawiony na wartość MONONE.

Jeśli określono GMLOGO, tylko niektóre komunikaty mogą zostać zwrócone przez następne wywołanie MQGET:

- Jeśli nie ma bieżącej grupy lub komunikatu logicznego, tylko komunikaty, które mają *MDSEQ* równe 1 i *MDOFF* równe 0 , są zakwalifikowane do zwrotu. W takiej sytuacji można użyć co najmniej jednej z następujących opcji, aby wybrać, który z zakwalifikowanych komunikatów jest zwracany:
  - MOMSGI
  - MOCORI
  - MOGRPI
- Jeśli istnieje bieżąca grupa lub komunikat logiczny, tylko następny komunikat w grupie lub następny segment w komunikacie logicznym jest zakwalifikowany do zwrotu i nie można go zmienić przez określenie opcji MO*.

W obu przypadkach opcje dopasowania, które nie mają zastosowania, mogą być nadal określone, ale wartość odpowiedniego pola w parametrze **MSGDSC** musi być zgodna z wartością odpowiedniego pola w komunikacie, który ma zostać zwrócony. Wywołanie nie powiedzie się z kodem przyczyny RC2247 , jeśli ten warunek nie zostanie spełniony.

Parametr *GMMO* jest ignorowany, jeśli określono wartość GMMUC lub GMBRWC.

Można podać co najmniej jedną z następujących opcji:

MOMSGI
Pobierz komunikat z określonym identyfikatorem komunikatu.

Ta opcja określa, że komunikat do pobrania musi mieć identyfikator komunikatu zgodny z wartością pola *MDMID* w parametrze **MSGDSC** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja nie zostanie podana, pole *MDMID* w parametrze **MSGDSC** zostanie zignorowane, a wszystkie identyfikatory komunikatów będą zgodne.

Uwaga: Identyfikator komunikatu MINONE jest wartością specjalną, która jest zgodna z dowolnym identyfikatorem komunikatu w deskryptorze MQMD dla komunikatu. Dlatego określenie parametru MOMSGI z parametrem MINONE jest takie samo, jak nieokreślenie parametru MOMSGI.

MOCORI
Pobierz komunikat o określonym identyfikatorze korelacji.

Ta opcja określa, że komunikat do pobrania musi mieć identyfikator korelacji zgodny z wartością pola *MDCID* w parametrze **MSGDSC** wywołania MQGET. To dopasowanie jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator komunikatu).

Jeśli ta opcja nie zostanie podana, pole *MDCID* w parametrze **MSGDSC** zostanie zignorowane, a wszystkie identyfikatory korelacji będą zgodne.

Uwaga: Identyfikator korelacji CINONE jest wartością specjalną, która jest zgodna z dowolnym identyfikatorem korelacji w strukturze MQMD dla komunikatu. Dlatego określenie parametru MOCORI z parametrem CINONE jest takie samo, jak nieokreślenie parametru MOCORI.

MOGRPI
Pobierz komunikat z określonym identyfikatorem grupy.

Ta opcja określa, że komunikat do pobrania musi mieć identyfikator grupy zgodny z wartością pola *MDGID* w parametrze **MSGDSC** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator korelacji).

Jeśli ta opcja nie zostanie podana, pole *MDGID* w parametrze **MSGDSC** zostanie zignorowane, a wszystkie identyfikatory grup będą zgodne.

Uwaga: Identyfikator grupy GINONE jest wartością specjalną, która jest zgodna z dowolnym identyfikatorem grupy w deskryptorze MQMD dla komunikatu. Dlatego określenie parametru MOGRPI z parametrem GINONE jest takie samo, jak nieokreślenie parametru MOGRPI.

MOSEQN
Pobierz komunikat o podanym numerze kolejnym komunikatu.

Ta opcja określa, że komunikat do pobrania musi mieć numer kolejny komunikatu zgodny z wartością pola *MDSEQ* w parametrze **MSGDSC** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład identyfikator grupy).

Jeśli ta opcja nie zostanie podana, pole *MDSEQ* w parametrze **MSGDSC** zostanie zignorowane, a wszystkie numery kolejne komunikatów będą zgodne.

MOOFFS
Pobierz komunikat z określonym przesunięciem.

Ta opcja określa, że komunikat do pobrania musi mieć przesunięcie zgodne z wartością pola *MDOFF* w parametrze **MSGDSC** wywołania MQGET. Ta zgodność jest uzupełnieniem wszystkich innych dopasowań, które mogą mieć zastosowanie (na przykład numeru kolejnego komunikatu).

Jeśli ta opcja nie zostanie podana, pole *MDOFF* w parametrze **MSGDSC** zostanie zignorowane, a wszystkie przesunięcia będą zgodne.

Jeśli nie określono żadnej z opisanych opcji, można użyć następującej opcji:

MONONE.
Brak dopasowań.

Ta opcja określa, że podczas wybierania komunikatu, który ma zostać zwrócony, nie mają być używane żadne dopasowania. Oznacza to, że wszystkie komunikaty w kolejce są zakwalifikowane do pobrania (ale podlegają kontroli przez opcje GMAMSA, GMASGA i GMCMPM).

Parametr MONONE jest zdefiniowany w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją MO*, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

To pole jest polem wejściowym. Wartością początkową tego pola jest MOMSGI z MOCORI. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER2.

Uwaga: Wartość początkowa pola *GMMO* jest definiowana w celu zachowania zgodności z menedżerami kolejek wcześniejszej wersji. Jednak podczas odczytywania serii komunikatów z kolejki bez użycia kryteriów wyboru ta wartość początkowa wymaga, aby aplikacja zresetować pola *MDMID* i *MDCID* do wartości MINONE i CINONE przed każdym wywołaniem MQGET. Aby uniknąć konieczności resetowania parametrów *MDMID* i *MDCID* , należy ustawić parametr *GMVER* na wartość GMVER2, a parametr *GMMO* na wartość MONONE.

GMOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje sterujące działaniem komendy MQGET.

Można podać co najmniej jedną z poniższych opcji. Jeśli wymagana jest więcej niż jedna wartość, można dodać wartości (nie należy dodawać tej samej stałej więcej niż raz). Podano niepoprawne kombinacje opcji; wszystkie pozostałe kombinacje są poprawne.

Opcje oczekiwania: Następujące opcje odnoszą się do oczekiwania na przybycie komunikatów do kolejki:

Znacznik GMWT
Poczekaj na nadejście komunikatu.

Aplikacja ma czekać, aż pojawi się odpowiedni komunikat. Maksymalny czas oczekiwania aplikacji jest określony w pliku *GMWI*.

Jeśli żądania MQGET są zablokowane lub żądania MQGET stają się zablokowane podczas oczekiwania, oczekiwanie jest anulowane, a wywołanie jest kończone z kodem CCFAIL i kodem przyczyny RC2016, niezależnie od tego, czy w kolejce znajdują się odpowiednie komunikaty.

Ta opcja może być używana z opcjami GMBRWF lub GMBRWN.

Jeśli kilka aplikacji oczekuje w tej samej kolejce współużytkowanej, aplikacja lub aplikacje, które są aktywowane po nadejściu odpowiedniego komunikatu, są opisane w dalszej części tej sekcji.

Uwaga: W poniższym opisie wywołanie MQGET przeglądania to takie, które określa jedną z opcji przeglądania, ale nie opcję GMLK. Wywołanie MQGET określające opcję GMLK jest traktowane jako wywołanie bez przeglądania.

- Jeśli oczekuje co najmniej jedno wywołanie MQGET bez przeglądania, ale nie oczekuje żadne wywołanie MQGET przeglądania, zostanie ono aktywowane.
- Jeśli co najmniej jedno wywołanie MQGET przeglądania oczekuje, ale nie ma żadnych wywołań MQGET przeglądania, wszystkie są aktywowane.
- Jeśli co najmniej jedno wywołanie MQGET bez przeglądania i co najmniej jedno wywołanie MQGET przeglądania jest oczekujące, jedno wywołanie MQGET bez przeglądania jest aktywowane i żadne, niektóre lub wszystkie wywołania MQGET przeglądania nie są aktywowane. (Nie można przewidzieć liczby aktywowanych wywołań MQGET przeglądania, ponieważ zależy to od kwestii związanych z planowaniem systemu operacyjnego i innych czynników).

Jeśli w tej samej kolejce oczekuje więcej niż jedno wywołanie MQGET bez przeglądania, aktywowane jest tylko jedno wywołanie. W takiej sytuacji menedżer kolejek próbuje nadać priorytet oczekującym wywołaniom bez przeglądania w następującej kolejności:

1. Konkretne żądania oczekiwania na pobranie, które mogą być spełnione tylko przez określone komunikaty, na przykład te z konkretnym *MDMID* lub *MDCID* (lub oba te komunikaty).
2. Ogólne żądania oczekiwania na pobranie, które mogą być spełnione przez dowolny komunikat.

Należy zwrócić uwagę na następujące kwestie:

- W pierwszej kategorii nie jest nadawany dodatkowy priorytet bardziej specyficznym żądaniom get-wait, na przykład tym, które określają zarówno *MDMID* , jak i *MDCID*.
- W żadnej z tych kategorii nie można przewidzieć, która aplikacja jest wybrana. W szczególności aplikacja oczekująca najdłużej nie musi być tą wybraną.

- Długość ścieżki i uwagi dotyczące planowania priorytetu w systemie operacyjnym mogą oznaczać, że oczekująca aplikacja o niższym priorytecie systemu operacyjnego niż oczekiwano pobiera komunikat.
- Może się również zdarzyć, że aplikacja, która nie oczekuje, pobierze komunikat w preferencjach do aplikacji, która go oczekuje.

Znacznik GMWT jest ignorowany, jeśli zostanie podany z parametrem GMBRWC lub GMMUC; nie jest zgłaszany żaden błąd.

GMNWT,
Wróć natychmiast, jeśli nie ma odpowiedniego komunikatu.

Aplikacja nie będzie czekać, jeśli nie będzie dostępny odpowiedni komunikat. Jest to przeciwieństwo opcji GMWT i jest ona definiowana w celu wspomagania dokumentacji programu. Jest to wartość domyślna, jeśli nie określono żadnej z nich.

GMFIQ,
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

Ta opcja wymusza niepowodzenie wywołania MQGET, jeśli menedżer kolejek jest w stanie wyciszania.

Jeśli ta opcja jest określona razem z tokenem GMWT, a oczekiwanie trwa w momencie, gdy menedżer kolejek przechodzi w stan wyciszania:

- Oczekiwanie zostało anulowane, a wywołanie zwraca kod zakończenia CCFAIL z kodem przyczyny RC2161 .

Jeśli parametr GMFIQ nie zostanie określony, a menedżer kolejek przejdzie w stan wyciszania, oczekiwanie nie zostanie anulowane.

Opcje punktu synchronizacji: Następujące opcje odnoszą się do udziału wywołania MQGET w jednostce pracy:

GMSYP
Pobierz komunikat z elementem sterującym punktu synchronizacji.

Żądanie ma działać w ramach zwykłych protokołów jednostki pracy. Komunikat jest oznaczany jako niedostępny dla innych aplikacji, ale jest usuwany z kolejki tylko wtedy, gdy jednostka pracy jest zatwierdzona. Komunikat zostanie ponownie udostępniony, jeśli jednostka pracy zostanie wycofana.

Jeśli ta opcja lub opcja GMNSYP nie jest określona, żądanie pobrania nie znajduje się w jednostce pracy.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF,
- GMBRWC
- GMBRWN
- GMLK,
- GMNSYP,
- Komenda GMPSYP
- GMUNLK

Komenda GMPSYP
Pobierz komunikat z elementem sterującym punktu synchronizacji, jeśli komunikat jest trwały.

Żądanie ma działać w ramach zwykłych protokołów jednostki pracy, ale tylko wtedy, gdy pobierany komunikat jest trwały. Komunikat trwały ma wartość PEPER w polu *MDPER* w strukturze MQMD.

- Jeśli komunikat jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby aplikacja miała określony parametr GMSYP.

- Jeśli komunikat nie jest trwały, menedżer kolejek przetwarza wywołanie tak, jakby aplikacja miała określony parametr GMNSYP (szczegółowe informacje zawiera poniższa sekcja).

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF,
- GMBRWC
- GMBRWN
- GMCMPM,
- GMNSYP,
- GMSYP
- GMUNLK

GMNSYP,
Pobierz komunikat bez sterowania punktem synchronizacji.

Żądanie ma działać poza normalnymi protokołami jednostki pracy. Komunikat jest natychmiast usuwany z kolejki (chyba że jest to żądanie przeglądania). Komunikat nie może zostać ponownie udostępniony przez wycofanie jednostki pracy.

Ta opcja jest używana, jeśli określono GMBRWF lub GMBRWN.

Jeśli ta opcja i opcja GMSYP nie są określone, żądanie pobrania nie znajduje się w jednostce pracy.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMSYP
- Komenda GMPSYP

Opcje przeglądania: Następujące opcje dotyczą przeglądania komunikatów w kolejce:

GMBRWF,
Przeglądaj od początku kolejki.

Gdy kolejka jest otwierana z opcją OOBRW, ustanawiany jest kursor przeglądania, umieszczony logicznie przed pierwszym komunikatem w kolejce. Kolejne wywołania MQGET określające opcję GMBRWF, GMBRWN lub GMBRWC mogą być używane do pobierania komunikatów z kolejki bez zniszczenia. Kursor przeglądania oznacza pozycję w obrębie komunikatów w kolejce, od której następne wywołanie MQGET z GMBRWN wyszukuje odpowiedni komunikat.

Wywołanie MQGET z GMBRWF powoduje zignorowanie poprzedniej pozycji kursora przeglądania. Pobierany jest pierwszy komunikat w kolejce, który spełnia warunki określone w deskryptorze komunikatu. Komunikat pozostaje w kolejce, a kursor przeglądania jest ustawiony na tym komunikacie.

Po tym wywołaniu kursor przeglądania jest umieszczany na zwróconym komunikacie. Jeśli komunikat zostanie usunięty z kolejki przed wykonaniem następnego wywołania MQGET z wartością GMBRWN, kursor przeglądania pozostanie na pozycji w kolejce, w której komunikat jest zajęty, nawet jeśli pozycja ta jest teraz pusta.

W razie potrzeby można użyć opcji GMMUC z wywołaniem MQGET bez przeglądania, aby usunąć komunikat z kolejki.

Kursor przeglądania nie jest przenoszony przez wywołanie MQGET bez przeglądania przy użyciu tego samego uchwytu *HOBJ* . Nie jest on również przenoszony przez wywołanie przeglądania MQGET, które zwraca kod zakończenia CCFAIL lub kod przyczyny RC2080 .

Razem z tą opcją można określić opcję GMLK, aby zablokować przeglądany komunikat.

Parametr GMBRWF można określić z dowolną poprawną kombinacją opcji GM* i MO*, które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli określono parametr GMLOGO, komunikaty są przeglądane w kolejności logicznej. Jeśli ta opcja zostanie pominięta, komunikaty będą przeglądane w kolejności fizycznej. Jeśli określono

wartość GMBRWF, można przełączać się między kolejnością logiczną i kolejnością fizyczną, ale kolejne wywołania MQGET używające wartości GMBRWN muszą przeglądać kolejkę w tej samej kolejności, co ostatnie wywołanie, które określało wartość GMBRWF dla uchwytu kolejki.

Informacje o grupach i segmentach, które menedżer kolejek przechowuje dla wywołań MQGET, które przeglądają komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które menedżer kolejek zachowuje dla wywołań MQGET, które usuwają komunikaty z kolejki. Jeśli określono GMBRWF, menedżer kolejek ignoruje informacje o grupach i segmentach podczas przeglądania i skanuje kolejkę tak, jakby nie było bieżącej grupy ani bieżącego komunikatu logicznego. Jeśli wywołanie MQGET powiedzie się (kod zakończenia CCOK lub CCWARN), informacje o grupie i segmencie do przeglądania są ustawiane na wartość zwracaną przez komunikat. Jeśli wywołanie nie powiedzie się, informacje o grupie i segmencie pozostają takie same, jak przed wywołaniem.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWC
- GMBRWN
- GMMUC
- GMSYP
- Komenda GMPSYP
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

GMBRWN
Przeglądaj z bieżącej pozycji w kolejce.

Kursor przeglądania jest przenoszony do następnego komunikatu w kolejce, który spełnia kryteria wyboru określone w wywołaniu MQGET. Komunikat jest zwracany do aplikacji, ale pozostaje w kolejce.

Po otwarciu kolejki do przeglądania pierwsze wywołanie przeglądania za pomocą uchwytu ma taki sam efekt, niezależnie od tego, czy określa opcję GMBRWF, czy GMBRWN.

Jeśli komunikat zostanie usunięty z kolejki przed wysłaniem następnego wywołania MQGET z wartością GMBRWN, kursor przeglądania pozostaje logicznie na pozycji w kolejce, w której komunikat jest zajęty, nawet jeśli pozycja ta jest teraz pusta.

Komunikaty są przechowywane w kolejce na jeden z dwóch sposobów:

- FIFO w ramach priorytetu (MSPRIO), lub
- FIFO bez względu na priorytet (MSFIFO)

Atrybut kolejki **MsgDeliverySequence** wskazuje, która metoda ma zastosowanie (szczegółowe informacje zawiera sekcja “Atrybuty kolejek” na stronie 1410).

Jeśli kolejka ma *MsgDeliverySequence* MSPRIO i komunikat pojawia się w kolejce o wyższym priorytecie niż ten wskazywany obecnie przez kursor przeglądania, komunikat ten nie zostanie znaleziony podczas bieżącego przeglądania kolejki za pomocą GMBRWN. Można go znaleźć tylko po zresetowaniu kursora przeglądania za pomocą GMBRWF (lub po ponownym otwarciu kolejki).

Opcja GMMUC może być później używana z wywołaniem MQGET bez przeglądania, jeśli jest to wymagane, w celu usunięcia komunikatu z kolejki.

Kursor przeglądania nie jest przenoszony przez wywołania MQGET bez przeglądania za pomocą tego samego uchwytu *HOBJ* .

Razem z tą opcją można określić opcję GMLK, aby zablokować przeglądany komunikat.

Wartość GMBRWN można określić z dowolną poprawną kombinacją opcji GM* i MO*, które sterują przetwarzaniem komunikatów w grupach i segmentach komunikatów logicznych.

Jeśli określono parametr GMLOGO, komunikaty są przeglądane w kolejności logicznej. Jeśli ta opcja zostanie pominięta, komunikaty będą przeglądane w kolejności fizycznej. Jeśli określono wartość GMBRWF, można przełączać się między kolejnością logiczną i kolejnością fizyczną, ale kolejne wywołania MQGET używające wartości GMBRWN muszą przeglądać kolejkę w tej samej kolejności, co ostatnie wywołanie, które określało wartość GMBRWF dla uchwytu kolejki. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2259 .

Uwaga: Należy zachować szczególną ostrożność, jeśli wywołanie MQGET jest używane do przeglądania poza końcem grupy komunikatów (lub komunikatu logicznego, który nie znajduje się w grupie), gdy nie określono parametru GMLOGO. Na przykład, jeśli ostatni komunikat w grupie występuje przed pierwszym komunikatem w grupie w kolejce, używając GMBRWN do przeglądania poza końcem grupy, podanie parametru MOSEQN z wartością *MDSEQ* ustawioną na 1 (w celu znalezienia pierwszego komunikatu w następnej grupie) spowoduje ponowne zwrócenie pierwszego komunikatu w grupie, która była już przeglądana. Może to nastąpić natychmiast lub kilka wywołań MQGET później (jeśli istnieją grupy pośredniczące).

Możliwość pętli nieskończonej można uniknąć, otwierając kolejkę dwukrotnie do przeglądania:

- Użyj pierwszego uchwytu, aby przeglądać tylko pierwszy komunikat w każdej grupie.
- Drugi uchwyt służy do przeglądania tylko komunikatów w konkretnej grupie.
- Użyj opcji MO*, aby przenieść drugi kursor przeglądania do pozycji pierwszego kursora przeglądania przed przeglądaniem komunikatów w grupie.
- Nie należy używać GMBRWN do przeglądania poza końcem grupy.

Informacje o grupach i segmentach, które menedżer kolejek przechowuje dla wywołań MQGET, które przeglądają komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które zachował dla wywołań MQGET, które usuwają komunikaty z kolejki.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF,
- GMBRWC
- GMMUC
- GMSYP
- Komenda GMPSYP
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

GMBRWC
Komunikat przeglądania pod kursorem przeglądania.

Ta opcja powoduje, że komunikat wskazywany przez kursor przeglądania jest pobierany bez zniszczenia, niezależnie od opcji MO* określonych w polu *GMMO* w MQGMO.

Komunikat wskazywany przez kursor przeglądania to ten, który został ostatnio pobrany przy użyciu opcji GMBRWF lub GMBRWN. Wywołanie nie powiedzie się, jeśli żadne z tych wywołań nie zostało wykonane dla tej kolejki od czasu jej otwarcia lub jeśli komunikat, który znajdował się pod kursorem przeglądania, został od tego czasu pobrany ze zniszczeniem.

Pozycja kursora przeglądania nie jest zmieniana przez to wywołanie.

W razie potrzeby można użyć opcji GMMUC z wywołaniem MQGET bez przeglądania, aby usunąć komunikat z kolejki.

Kursor przeglądania nie jest przenoszony przez wywołanie MQGET bez przeglądania przy użyciu tego samego uchwytu *HOBJ* . Nie jest on również przenoszony przez wywołanie przeglądania MQGET, które zwraca kod zakończenia CCFAIL lub kod przyczyny RC2080.

Jeśli dla parametru GMLK określono wartość GMBRWC:

- Jeśli komunikat jest już zablokowany, musi to być ten, który znajduje się pod kursorem, tak aby był zwracany bez odblokowywania i ponownego blokowania; komunikat pozostaje zablokowany.
- Jeśli nie ma zablokowanego komunikatu, komunikat pod kursorem przeglądania (jeśli istnieje) jest zablokowany i zwracany do aplikacji; jeśli nie ma komunikatu pod kursorem przeglądania, wywołanie nie powiedzie się.

Jeśli wartość GMBRWC jest określona bez wartości GMLK:

- Jeśli komunikat jest już zablokowany, musi to być ten, który znajduje się pod kursorem. Ten komunikat jest zwracany do aplikacji, a następnie odblokowywany. Ponieważ komunikat jest teraz odblokowany, nie ma gwarancji, że może zostać ponownie przejrzany lub pobrany destrukcyjnie (może zostać pobrany destrukcyjnie przez inną aplikację pobierającą komunikaty z kolejki).
- Jeśli nie ma zablokowanego komunikatu, komunikat pod kursorem przeglądania (jeśli istnieje) jest zwracany do aplikacji; jeśli nie ma komunikatu pod kursorem przeglądania, wywołanie nie powiedzie się.

Jeśli właściwość GMCMPM została określona z parametrem GMBRWC, kursor przeglądania musi identyfikować komunikat z polem *MDOFF* w deskryptorze MQMD o wartości zero. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2246 .

Informacje o grupach i segmentach, które menedżer kolejek przechowuje dla wywołań MQGET, które przeglądają komunikaty w kolejce, są oddzielone od informacji o grupach i segmentach, które zachował dla wywołań MQGET, które usuwają komunikaty z kolejki.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF,
- GMBRWN
- GMMUC
- GMSYP
- Komenda GMPSYP
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta do przeglądania.

GMMUC
Pobierz komunikat pod kursorem przeglądania.

Ta opcja powoduje pobranie komunikatu wskazywanego przez kursor przeglądania, niezależnie od opcji MO* określonych w polu *GMMO* w MQGMO. Komunikat jest usuwany z kolejki.

Komunikat wskazywany przez kursor przeglądania to ten, który został ostatnio pobrany przy użyciu opcji GMBRWF lub GMBRWN.

Jeśli GMCMPM określono z GMMUC, kursor przeglądania musi identyfikować komunikat z zerowym polem *MDOFF* w MQMD. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2246 .

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMBRWF,
- GMBRWC
- GMBRWN
- GMUNLK

Jest to również błąd, jeśli kolejka nie została otwarta zarówno do przeglądania, jak i do wprowadzania. Jeśli kursor przeglądania nie wskazuje aktualnie na komunikat, który można pobrać, wywołanie MQGET zwraca błąd.

Opcje blokowania: Następujące opcje dotyczą blokowania komunikatów w kolejce:

GMLK,
Komunikat blokady.

Ta opcja blokuje przeglądany komunikat, dzięki czemu komunikat staje się niewidoczny dla każdego innego uchwytu otwartego dla kolejki. Opcję można podać tylko wtedy, gdy podano jedną z następujących opcji:

- GMBRWF,
- GMBRWN
- GMBRWC

Dla każdego uchwytu kolejki może być zablokowany tylko jeden komunikat, ale może to być komunikat logiczny lub komunikat fizyczny:

- Jeśli określono GMCMPM, wszystkie segmenty komunikatów, które tworzą komunikat logiczny, są zablokowane w uchwycie kolejki (jeśli wszystkie znajdują się w kolejce i są dostępne do pobrania).
- Jeśli parametr GMCMPM nie jest określony, tylko pojedynczy komunikat fizyczny jest blokowany dla uchwytu kolejki. Jeśli ten komunikat jest segmentem komunikatu logicznego, zablokowany segment uniemożliwia pobranie lub przeglądanie komunikatu logicznego innym aplikacjom używającym GMCMPM.

Zablokowany komunikat jest zawsze tym, który znajduje się pod kursorem przeglądania, a komunikat można usunąć z kolejki za pomocą późniejszego wywołania MQGET, które określa opcję GMMUC. Inne wywołania MQGET używające uchwytu kolejki mogą również usunąć komunikat (na przykład wywołanie, które określa identyfikator zablokowanego komunikatu).

Jeśli wywołanie zwróci kod zakończenia CCFAIL lub CCWARN z kodem przyczyny RC2080, komunikat nie zostanie zablokowany.

Jeśli aplikacja nie usunie komunikatu z kolejki, blokada zostanie zwolniona przez:

- Wykonanie kolejnego wywołania MQGET dla tego uchwytu, z określoną wartością GMBRWF lub GMBRWN (z wartością GMLK lub bez niej); komunikat zostanie odblokowany, jeśli wywołanie zakończy się z wartością CCOK lub CCWARN, ale pozostanie zablokowane, jeśli wywołanie zakończy się z wartością CCFAIL. Istnieją jednak wyjątki:
  - Komunikat nie jest odblokowywany, jeśli komenda CCWARN zostanie zwrócona z kodem powrotu RC2080.
  - Komunikat zostanie odblokowany, jeśli komenda CCFAIL zostanie zwrócona z kodem powrotu RC2033.

  Jeśli określono również wartość GMLK, zwracany komunikat jest zablokowany. Jeśli nie określono parametru GMLK, po wywołaniu nie ma zablokowanego komunikatu.

  Jeśli określono GMWT i żaden komunikat nie jest natychmiast dostępny, odblokowanie oryginalnego komunikatu następuje przed rozpoczęciem oczekiwania (pod warunkiem, że wywołanie jest wolne od błędu).
- Wywołanie innego wywołania MQGET dla tego uchwytu z użyciem GMBRWC (bez GMLK); komunikat jest odblokowywany, jeśli wywołanie zostanie zakończone z użyciem CCOK lub CCWARN, ale pozostaje zablokowane, jeśli wywołanie zostanie zakończone z użyciem CCFAIL. Jednak zastosowanie ma następujący wyjątek:
  - Komunikat nie jest odblokowywany, jeśli komenda CCWARN zostanie zwrócona z kodem powrotu RC2080.
- Wysyłanie kolejnego wywołania MQGET dla tego uchwytu za pomocą komendy GMUNLK.
- Wywołanie MQCLOSE dla tego uchwytu (jawnie lub niejawnie przez zakończenie aplikacji).

Do określenia tej opcji nie jest wymagana żadna specjalna opcja otwierania, inna niż OOBRW, która jest wymagana do określenia towarzyszącej opcji przeglądania.

Ta opcja nie jest poprawna z żadną z następujących opcji:

- GMSYP
- Komenda GMPSYP
- GMUNLK

GMUNLK
Odblokuj komunikat.

Komunikat, który ma zostać odblokowany, musi być wcześniej zablokowany przez wywołanie MQGET z opcją GMLK. Jeśli dla tego uchwytu nie ma zablokowanego komunikatu, wywołanie zostanie zakończone z CCWARN i RC2209 .

Parametry **MSGDSC**, **BUFLEN**, **BUFFER**i **DATLEN** nie są sprawdzane ani zmieniane, jeśli podano parametr GMUNLK. W pliku *BUFFER*nie jest zwracany żaden komunikat.

Do określenia tej opcji nie jest wymagana żadna specjalna opcja otwarcia (chociaż do wydania żądania blokady w pierwszej kolejności konieczne jest użycie funkcji OOBRW).

Ta opcja nie jest poprawna dla żadnych opcji z wyjątkiem następujących:

- GMNWT,
- GMNSYP,

Obie te opcje są przyjmowane niezależnie od tego, czy zostały określone, czy nie.

Opcje danych komunikatu: Następujące opcje odnoszą się do przetwarzania danych komunikatu, gdy komunikat jest odczytywany z kolejki:

GMATM,
Zezwalaj na obcięcie danych komunikatu.

Jeśli bufor komunikatów jest zbyt mały, aby pomieścić cały komunikat, ta opcja umożliwia wywołanie MQGET zapełnienie buforu tak dużą ilością komunikatu, jaka może być wstrzymana przez bufor, wysłanie kodu zakończenia ostrzeżenia i zakończenie jego przetwarzania. Oznacza to:

- Podczas przeglądania komunikatów kursor przeglądania jest przenoszony do zwróconego komunikatu.
- Podczas usuwania komunikatów zwracany komunikat jest usuwany z kolejki.
- Jeśli nie wystąpi inny błąd, zwracany jest kod przyczyny RC2079 .

Bez tej opcji bufor jest nadal wypełniany komunikatem, który może być wstrzymany, generowany jest kod zakończenia ostrzeżenia, ale przetwarzanie nie jest zakończone. Oznacza to:

- Podczas przeglądania komunikatów kursor przeglądania nie jest zaawansowany.
- Podczas usuwania komunikatów komunikat nie jest usuwany z kolejki.
- Jeśli nie wystąpi inny błąd, zwracany jest kod przyczyny RC2080 .

GMCONV
Konwertuj dane komunikatu.

Ta opcja żąda, aby dane aplikacji w komunikacie zostały przekształcone w celu zapewnienia zgodności z wartościami *MDCSI* i *MDENC* określonymi w parametrze **MSGDSC** wywołania MQGET przed skopiowaniem danych do parametru **BUFFER** .

Pole *MDFMT* określone podczas umieszczania komunikatu jest przyjmowane przez proces konwersji w celu zidentyfikowania rodzaju danych w komunikacie. Konwersja danych komunikatu jest wykonywana przez menedżer kolejek dla formatów wbudowanych oraz przez program zewnętrzny napisany przez użytkownika dla innych formatów.

- Jeśli konwersja zakończy się pomyślnie, pola *MDCSI* i *MDENC* określone w parametrze **MSGDSC** pozostaną niezmienione po powrocie z wywołania MQGET.
- Jeśli konwersja nie może zostać wykonana pomyślnie (ale wywołanie MQGET w przeciwnym razie zakończy się bez błędu), dane komunikatu są zwracane bez konwersji, a pola *MDCSI*

i *MDENC* w pliku *MSGDSC* są ustawiane na wartości dla nieprzekształconego komunikatu. W tym przypadku kod zakończenia to CCWARN.

W obu tych przypadkach pola te opisują identyfikator zestawu znaków i kodowanie danych komunikatu, które są zwracane w parametrze **BUFFER** .

Listę nazw formatów, dla których menedżer kolejek wykonuje konwersję, zawiera pole *MDFMT* opisane w sekcji "MQMD (deskryptor komunikatu) w systemie IBM i" na stronie 1141 .

Opcje grup i segmentów: Następujące opcje odnoszą się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Poniższe definicje mogą pomóc w zrozumieniu opcji:

Komunikat fizyczny
Jest to najmniejsza jednostka informacji, którą można umieścić w kolejce lub z niej usunąć. Często odpowiada ona informacjom określonym lub pobranym w pojedynczym wywołaniu MQPUT, MQPUT1lub MQGET. Każdy komunikat fizyczny ma własny deskryptor komunikatu (MQMD). Zazwyczaj komunikaty fizyczne są rozróżniane na podstawie różnych wartości identyfikatora komunikatu (pole*MDMID* w strukturze MQMD), chociaż nie jest to wymuszane przez menedżer kolejek.

Komunikat logiczny
Jest to pojedyncza jednostka informacji o aplikacji. W przypadku braku ograniczeń systemowych komunikat logiczny byłby taki sam, jak komunikat fizyczny. Jeśli jednak komunikaty logiczne są duże, ograniczenia systemowe mogą spowodować, że wskazane lub konieczne będzie podzielenie komunikatu logicznego na dwa lub więcej komunikatów fizycznych, zwanych segmentami.

Komunikat logiczny, który został podzielony na segmenty, składa się z dwóch lub większej liczby komunikatów fizycznych, które mają ten sam identyfikator grupy o wartości innej niż NULL (pole*MDGID* w strukturze MQMD) i ten sam numer kolejny komunikatu (pole*MDSEQ* w strukturze MQMD). Segmenty są rozróżniane przez różne wartości przesunięcia segmentu (pole*MDOFF* w strukturze MQMD), które powoduje przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dozwolona przez aplikację wysyłającą, ma również identyfikator grupy o wartości innej niż NULL, chociaż w tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają pusty identyfikator grupy (GINONE), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy
Jest to zestaw co najmniej jednego komunikatu logicznego, który ma ten sam identyfikator grupy o wartości innej niż NULL. Komunikaty logiczne w grupie są rozróżniane przez różne wartości numeru kolejnego komunikatu, który jest liczbą całkowitą z zakresu od 1 do n, gdzie n jest liczbą komunikatów logicznych w grupie. Jeśli jeden lub więcej komunikatów logicznych jest segmentowanych, w grupie znajduje się więcej niż n komunikatów fizycznych.

GLOGO
Komunikaty w grupach i segmentach komunikatów logicznych są zwracane w kolejności logicznej.

Ta opcja steruje kolejnością zwracania komunikatów przez kolejne wywołania MQGET dla uchwytu kolejki. Opcja musi być określona w każdym z tych wywołań, aby uzyskać efekt.

Jeśli parametr GMLOGO jest określony dla kolejnych wywołań MQGET dla uchwytu kolejki, komunikaty w grupach są zwracane w kolejności określonej przez numery kolejne komunikatów, a segmenty komunikatów logicznych są zwracane w kolejności określonej przez przesunięcia segmentów. Ta kolejność może różnić się od kolejności, w jakiej te komunikaty i segmenty występują w kolejce.

Uwaga: Określenie GMLOGO nie ma negatywnego wpływu na komunikaty, które nie należą do grup i nie są segmentami. W rezultacie takie komunikaty są traktowane tak, jakby każdy należał do grupy komunikatów składającej się tylko z jednego komunikatu. W związku z tym podczas pobierania komunikatów z kolejek, które mogą zawierać mieszankę komunikatów w grupach,

segmentach komunikatów i niesegmentowanych komunikatach, które nie są w grupach, można bezpiecznie określić parametr GMLOGO.

Aby zwrócić komunikaty w wymaganej kolejności, menedżer kolejek zachowuje informacje o grupie i segmencie między kolejnymi wywołaniami MQGET. Informacje te identyfikują bieżącą grupę komunikatów i bieżący komunikat logiczny dla uchwytu kolejki, bieżącą pozycję w grupie i komunikat logiczny oraz informację, czy komunikaty są wczytywane w ramach jednostki pracy. Ponieważ menedżer kolejek zachowuje te informacje, aplikacja nie musi ustawiać informacji o grupie i segmencie przed każdym wywołaniem MQGET. W szczególności oznacza to, że aplikacja nie musi ustawiać pól *MDGID*, *MDSEQ*i *MDOFF* w strukturze MQMD. Jednak aplikacja musi poprawnie ustawić opcję GMSYP lub GMNSYP w każdym wywołaniu.

Po otwarciu kolejki nie ma bieżącej grupy komunikatów ani bieżącego komunikatu logicznego. Grupa komunikatów staje się bieżącą grupą komunikatów, gdy komunikat z flagą MFMIG jest zwracany przez wywołanie MQGET. Jeśli w kolejnych wywołaniach określono GMLOGO, grupa ta pozostaje bieżącą grupą do momentu zwrócenia komunikatu, który zawiera:

- MFLMIG bez MFSEG (czyli ostatni komunikat logiczny w grupie nie jest segmentowany), lub
- MFLMIG z MFLSEG (to znaczy, zwrócony komunikat jest ostatnim segmentem ostatniego komunikatu logicznego w grupie).

Po zwróceniu takiego komunikatu grupa komunikatów jest przerywana, a po pomyślnym zakończeniu wywołania MQGET nie ma już bieżącej grupy. W podobny sposób komunikat logiczny staje się bieżącym komunikatem logicznym, gdy komunikat z flagą MFSEG jest zwracany przez wywołanie MQGET, a komunikat logiczny jest przerywany po zwróceniu komunikatu z flagą MFLSEG.

Jeśli nie określono żadnych kryteriów wyboru, kolejne wywołania MQGET zwracają (w poprawnej kolejności) komunikaty dla pierwszej grupy komunikatów w kolejce, następnie komunikaty dla drugiej grupy komunikatów itd. do czasu, aż nie będzie dostępnych więcej komunikatów. Można wybrać konkretne grupy komunikatów, podając jedną lub więcej spośród następujących opcji w polu *GMMO* :

- MOMSGI
- MOCORI
- MOGRPI

Jednak te opcje są dostępne tylko wtedy, gdy nie ma bieżącej grupy komunikatów ani komunikatu logicznego. Patrz pole *GMMO* opisane w tym temacie.

Tabela 702 na stronie 1120 przedstawia wartości pól *MDMID*, *MDCID*, *MDGID*, *MDSEQ*i *MDOFF* , które są wyszukiwane przez menedżer kolejek podczas próby znalezienia komunikatu zwracanego w wywołaniu MQGET. Dotyczy to zarówno usuwania komunikatów z kolejki, jak i przeglądania komunikatów w kolejce. Kolumny w tabeli mają następujące znaczenie:

SŁOWO PROTOKOŁU
Wskazuje, czy opcja GMLOGO jest określona w wywołaniu.

Cur grp
Wskazuje, czy przed wywołaniem istnieje bieżąca grupa komunikatów.

Komunikat dziennika Cur
Wskazuje, czy przed wywołaniem istnieje bieżący komunikat logiczny.

Inne kolumny
Pokaż wartości, których szuka menedżer kolejek. "Poprzedni" oznacza wartość zwróconą dla pola w poprzednim komunikacie dla uchwytu kolejki.

Tabela 702. Opcje MQGET dotyczące komunikatów w grupach i segmentach komunikatów logicznych							
Określone opcje	**Status grupy i komunikatu dziennika przed wywołaniem**		**Wartości, których szuka menedżer kolejek**				
SŁOWO PROTOKOŁU	**Bieżąca grupa**	**Bieżące komunikaty dziennika**	*MDMID*	*MDCID*	*MDGID*	*MDSEQ*	*MDOFF*
Tak	Nie	Nie	Kontrolowane przez *GMMO*	Kontrolowane przez *GMMO*	Kontrolowane przez *GMMO*	1	0
Tak	Nie	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + poprzednia długość segmentu
Tak	Tak	Nie	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Dowolny identyfikator komunikatu	Dowolny identyfikator korelacji	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + poprzednia długość segmentu
Nie	Albo	Albo	Kontrolowane przez *GMMO*	Kontrolowane przez *GMMO*	Kontrolowane przez *GMMO*	Kontrolowane przez *GMMO*	Kontrolowane przez *GMMO*

Jeśli w kolejce znajduje się wiele grup komunikatów, które mogą zostać zwrócone, grupy są zwracane w kolejności określonej przez pozycję w kolejce pierwszego segmentu pierwszego komunikatu logicznego w każdej grupie (tzn. komunikaty fizyczne, które mają numer kolejny komunikatu równy 1 i przesunięcia równe 0, określają kolejność, w jakiej zwracane są odpowiednie grupy).

Opcja GMLOGO wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy logiczny komunikat lub segment w grupie jest pobierany w ramach jednostki pracy, wszystkie inne logiczne komunikaty i segmenty w grupie muszą zostać pobrane w ramach jednostki pracy, jeśli używany jest ten sam uchwyt kolejki. Nie muszą one jednak być pobierane w ramach tej samej jednostki pracy. Umożliwia to podzielenie grupy komunikatów składającej się z wielu komunikatów fizycznych na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.
- Jeśli pierwszy logiczny komunikat lub segment w grupie nie jest pobierany w ramach jednostki pracy, żaden z pozostałych logicznych komunikatów i segmentów w grupie nie może zostać pobrany w ramach jednostki pracy, jeśli używany jest ten sam uchwyt kolejki.

Jeśli te warunki nie są spełnione, wywołanie MQGET kończy się niepowodzeniem z kodem przyczyny RC2245 .

Jeśli określono GMLOGO, wartość MQGMO podana w wywołaniu MQGET nie może być mniejsza niż GMVER2, a wartość MQMD nie może być mniejsza niż MDVER2. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2256 lub RC2257 (w zależności od przypadku).

Jeśli nie określono parametru GMLOGO dla kolejnych wywołań MQGET dla uchwytu kolejki, komunikaty są zwracane bez względu na to, czy należą do grup komunikatów, czy też są segmentami komunikatów logicznych. Oznacza to, że komunikaty lub segmenty z konkretnej grupy lub komunikatu logicznego mogą być zwracane w nieodpowiedniej kolejności lub mogą być mieszane z komunikatami lub segmentami z innych grup lub komunikatów logicznych albo z komunikatami, które nie są w grupach i nie są segmentami. W tej sytuacji konkretne komunikaty zwracane przez kolejne wywołania MQGET są kontrolowane przez opcje MO* określone w tych wywołaniach (szczegółowe informacje na temat tych opcji zawiera pole *GMMO* opisane w sekcji “MQGMO (opcje pobierania komunikatów) w systemie IBM i” na stronie 1106).

Jest to technika, która może być używana do restartowania grupy komunikatów lub komunikatu logicznego w środku, po wystąpieniu awarii systemu. Po zrestartowaniu systemu aplikacja może ustawić odpowiednie wartości w polach *MDGID*, *MDSEQ*, *MDOFF*i *GMMO* , a następnie uruchomić wywołanie MQGET z opcją GMSYP lub GMNSYP, jeśli jest to konieczne, ale bez określania parametru GMLOGO. Jeśli to wywołanie powiedzie się, menedżer kolejek zachowuje informacje o grupie i segmencie, a kolejne wywołania MQGET używające tego uchwytu kolejki mogą określać wartość GMLOGO jako normalną.

Informacje o grupie i segmencie, które menedżer kolejek przechowuje dla wywołania MQGET, są oddzielone od informacji o grupie i segmencie, które przechowuje dla wywołania MQPUT. Ponadto menedżer kolejek zachowuje osobne informacje dla:

- Wywołania MQGET usuwające komunikaty z kolejki.
- Wywołania MQGET, które przeglądają komunikaty w kolejce.

Dla dowolnego uchwytu kolejki aplikacja może swobodnie łączyć wywołania MQGET, które określają wywołania GMLOGO z wywołaniami MQGET, które tego nie robią, ale należy zauważyć następujące punkty:

- Jeśli parametr GMLOGO nie zostanie określony, każde pomyślne wywołanie MQGET spowoduje, że menedżer kolejek ustawi informacje o zapisanej grupie i segmencie na wartości odpowiadające zwróconemu komunikatowi; spowoduje to zastąpienie istniejących informacji o grupie i segmencie zachowanych przez menedżer kolejek dla uchwytu kolejki. Modyfikowane są tylko informacje odpowiednie dla działania wywołania (przeglądanie lub usuwanie).
- Jeśli nie określono parametru GMLOGO, wywołanie nie kończy się niepowodzeniem, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może jednak zakończyć się powodzeniem z kodem zakończenia CCWARN. Tabela 703 na stronie 1121 przedstawia różne obserwacje, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia jest inny niż CCOK, kod przyczyny jest jednym z następujących:
  - RC2241
  - RC2242
  - RC2245

  Uwaga: Menedżer kolejek nie sprawdza informacji o grupie i segmencie podczas przeglądania kolejki lub podczas zamykania kolejki, która została otwarta do przeglądania, ale nie została wprowadzona. W takich przypadkach kodem zakończenia jest zawsze CCOK (przy założeniu, że nie ma innych błędów).

Tabela 703. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupie i segmencie

Bieżące połączenie to	Poprzednie wywołanie to MQGET z GMLOGO	Poprzednie wywołanie to MQGET bez GMLOGO
MQGET z GMLOGO	CCFAIL (poczta elektroniczna)	CCFAIL (poczta elektroniczna)
MQGET bez GMLOGO	CWARN	CKOK

Tabela 703. Wynik, gdy wywołanie MQGET lub MQCLOSE nie jest spójne z informacjami o grupie i segmencie (kontynuacja)		
Bieżące połączenie to	**Poprzednie wywołanie to MQGET z GMLOGO**	**Poprzednie wywołanie to MQGET bez GMLOGO**
MQCLOSE z niezakończoną grupą lub komunikatem logicznym	CWARN	CKOK

Aplikacje, które po prostu chcą pobierać komunikaty i segmenty w porządku logicznym, są zalecane do określenia parametru GMLOGO, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia z konieczności zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tymi informacjami. Jednak wyspecjalizowane aplikacje mogą wymagać większej kontroli niż ta, która jest udostępniana przez opcję GMLOGO, i można to osiągnąć, nie określając tej opcji. W takim przypadku aplikacja musi upewnić się, że pola *MDMID*, *MDCID*, *MDGID*, *MDSEQ*i *MDOFF* w MQMD oraz opcje MO* w *GMMO* w MQGMO są ustawione poprawnie przed każdym wywołaniem MQGET.

Na przykład aplikacja, która chce przekazywać odbierane komunikaty fizyczne, bez względu na to, czy znajdują się one w grupach, czy w segmentach komunikatów logicznych, nie powinna określać parametru GMLOGO. Jest to spowodowane tym, że w złożonej sieci z wieloma ścieżkami między wysyłającymi i odbierającymi menedżerami kolejek komunikaty fizyczne mogą pojawiać się w nieodpowiedniej kolejności. Jeśli w wywołaniu MQPUT nie zostanie określona wartość GMLOGO i odpowiadająca jej wartość PMLOGO, aplikacja przekazująca może pobrać i przekazać każdy komunikat fizyczny natychmiast po jego nadejściu, bez konieczności oczekiwania na następny komunikat w kolejności logicznej.

W odpowiednich okolicznościach można określić GMLOGO z dowolną inną opcją GM* i z różnymi opcjami MO*.

GMCMPM,
Można pobierać tylko kompletne komunikaty logiczne.

Ta opcja określa, że wywołanie MQGET może zwrócić tylko pełny komunikat logiczny. Jeśli komunikat logiczny jest segmentowany, menedżer kolejek składa segmenty i zwraca do aplikacji pełny komunikat logiczny. Fakt, że komunikat logiczny został segmentowany, nie jest widoczny dla aplikacji, która go pobiera.

Uwaga: Jest to jedyna opcja, która powoduje, że menedżer kolejek składa segmenty komunikatów. Jeśli ta opcja nie zostanie określona, segmenty będą zwracane indywidualnie do aplikacji, jeśli są obecne w kolejce (i spełniają inne kryteria wyboru określone w wywołaniu MQGET). Aplikacje, które nie chcą otrzymywać pojedynczych segmentów, powinny zawsze określać GMCMPM.

Aby użyć tej opcji, aplikacja musi udostępnić bufor, który jest wystarczająco duży, aby pomieścić cały komunikat, lub określić opcję GMATM.

Jeśli kolejka zawiera segmentowane komunikaty z brakującymi segmentami (na przykład dlatego, że zostały opóźnione w sieci i jeszcze nie dotarły), podanie parametru GMCMPM uniemożliwia pobranie segmentów należących do niekompletnych komunikatów logicznych. Jednak te segmenty komunikatów nadal mają wpływ na wartość atrybutu kolejki **CurrentQDepth** . Oznacza to, że mogą nie istnieć żadne możliwe do pobrania komunikaty logiczne, nawet jeśli wartość *CurrentQDepth* jest większa od zera.

W przypadku trwałych komunikatów menedżer kolejek może złożyć segmenty tylko w ramach jednostki pracy:

- Jeśli wywołanie MQGET działa w obrębie jednostki pracy zdefiniowanej przez użytkownika, używana jest ta jednostka pracy. Jeśli wywołanie zakończy się niepowodzeniem przez proces ponownego składania, menedżer kolejek przywraca do kolejki wszystkie segmenty, które zostały

usunięte podczas ponownego składania. Jednak niepowodzenie nie uniemożliwia pomyślnego zatwierdzenia jednostki pracy.

- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika i nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy tylko na czas trwania wywołania. Jeśli wywołanie powiedzie się, menedżer kolejek automatycznie zatwierdza jednostkę pracy (aplikacja nie musi tego robić). Jeśli wywołanie nie powiedzie się, menedżer kolejek wycofa jednostkę pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika, ale jednostka pracy zdefiniowana przez użytkownika istnieje, menedżer kolejek nie może wykonać ponownego składania. Jeśli komunikat nie wymaga ponownego składania, wywołanie nadal może zakończyć się powodzeniem. Jeśli jednak komunikat wymaga ponownego złożenia, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2255 .

W przypadku nietrwałych komunikatów menedżer kolejek nie wymaga, aby jednostka pracy była dostępna w celu przeprowadzenia reasemblacji.

Każdy komunikat fizyczny, który jest segmentem, ma własny deskryptor komunikatu. W przypadku segmentów tworzących pojedynczy komunikat logiczny większość pól w deskryptorze komunikatu jest taka sama dla wszystkich segmentów w komunikacie logicznym-zwykle są to tylko pola *MDMID*, *MDOFF*i *MDMFL* , które różnią się między segmentami w komunikacie logicznym. Jeśli jednak segment zostanie umieszczony w kolejce niedostarczonych komunikatów w pośrednim menedżerze kolejek, program obsługi DLQ pobiera komunikat z opcją GMCONV, co może spowodować zmianę zestawu znaków lub kodowania segmentu. Jeśli program obsługi DLQ pomyślnie wyśle segment w swoją stronę, segment może mieć zestaw znaków lub kodowanie różniące się od innych segmentów w komunikacie logicznym, gdy segment w końcu dotrze do docelowego menedżera kolejek.

Menedżer kolejek nie może ponownie złożyć komunikatu logicznego składającego się z segmentów, w których pola *MDCSI*, *MDENC*lub oba różnią się między sobą, w jeden komunikat logiczny. Zamiast tego menedżer kolejek składa i zwraca pierwsze kilka kolejnych segmentów na początku komunikatu logicznego, które mają takie same identyfikatory zestawu znaków i kodowania, a wywołanie MQGET kończy działanie z kodem zakończenia CCWARN i kodem przyczyny RC2243 lub RC2244 (w zależności od przypadku). Dzieje się tak niezależnie od tego, czy określono parametr GMCONV. Aby pobrać pozostałe segmenty, aplikacja musi ponownie uruchomić wywołanie MQGET bez opcji GMCMPM, pobierając segmenty jeden po drugim. Komenda GMLOGO może być używana do pobierania pozostałych segmentów w kolejności.

Możliwe jest również, że aplikacja, która umieszcza segmenty w celu ustawienia innych pól w deskryptorze komunikatu na wartości różniące się między segmentami. Nie ma jednak żadnej korzyści, jeśli aplikacja odbierająca używa modułu GMCMPM do pobrania komunikatu logicznego. Gdy menedżer kolejek ponownie składa komunikat logiczny, zwraca w deskryptorze komunikatu wartości z deskryptora komunikatu dla pierwszego segmentu. Jedynym wyjątkiem jest pole *MDMFL* , które menedżer kolejek ustawia w celu wskazania, że złożony komunikat jest jedynym segmentem.

Jeśli dla komunikatu raportu określono GMCMPM, menedżer kolejek wykonuje przetwarzanie specjalne. Menedżer kolejek sprawdza kolejkę, aby sprawdzić, czy wszystkie komunikaty raportu tego typu dotyczące różnych segmentów w komunikacie logicznym znajdują się w kolejce. Jeśli tak, można je pobrać jako pojedynczy komunikat, podając GMCMPM. Aby było to możliwe, komunikaty raportu muszą zostać wygenerowane przez menedżer kolejek lub agent MCA, który obsługuje segmentację, albo aplikacja inicjująca musi zażądać co najmniej 100 bajtów danych komunikatu (należy określić odpowiednie opcje RO* D lub RO* F). Jeśli dla segmentu występuje mniej niż pełna ilość danych aplikacji, brakujące bajty są zastępowane przez wartości null w zwróconym komunikacie raportu.

Jeśli GMCMPM określono z GMMUC lub GMBRWC, kursor przeglądania musi być ustawiony na komunikacie z polem *MDOFF* w MQMD o wartości 0. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2246 .

GMCMPM implikuje GMASGA, który nie musi być określony.

Opcję GMCMPM można podać z dowolną inną opcją GM* oprócz opcji GMPSYP oraz z dowolną z opcji MO* z wyjątkiem opcji MOOFFS.

GMAMSA,
Wszystkie komunikaty w grupie muszą być dostępne.

Ta opcja określa, że komunikaty w grupie będą dostępne do pobrania tylko wtedy, gdy wszystkie komunikaty w grupie będą dostępne. Jeśli kolejka zawiera grupy komunikatów z brakującymi niektórymi komunikatami (na przykład dlatego, że zostały one opóźnione w sieci i jeszcze nie dotarły), podanie parametru GMAMSA uniemożliwia pobranie komunikatów należących do niekompletnych grup. Jednak te komunikaty nadal mają wpływ na wartość atrybutu kolejki **CurrentQDepth** . Oznacza to, że grupy komunikatów, które można pobrać, mogą nie istnieć, nawet jeśli wartość **CurrentQDepth** jest większa od zera. Jeśli nie ma innych komunikatów, które można pobrać, kod przyczyny RC2033 jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie GMAMSA zależy od tego, czy określono również GMLOGO:

- Jeśli zostaną podane obie opcje, działanie GMAMSA będzie miało wpływ tylko na brak bieżącej grupy lub komunikatu logicznego. Jeśli istnieje bieżąca grupa lub komunikat logiczny, wartość GMAMSA jest ignorowana. Oznacza to, że GMAMSA może pozostać włączony podczas przetwarzania komunikatów w porządku logicznym.
- Jeśli GMAMSA jest określony bez GMLOGO, GMAMSA zawsze daje efekt. Oznacza to, że opcja musi zostać wyłączona po usunięciu pierwszego komunikatu z grupy z kolejki, aby możliwe było usunięcie pozostałych komunikatów z grupy.

Pomyślne zakończenie wywołania MQGET z określeniem wartości GMAMSA oznacza, że w momencie wywołania MQGET wszystkie komunikaty w grupie znajdowały się w kolejce. Należy jednak pamiętać, że inne aplikacje nadal mogą usuwać komunikaty z grupy (grupa nie jest zablokowana dla aplikacji, która pobiera pierwszy komunikat z grupy).

Jeśli ta opcja nie zostanie podana, komunikaty należące do grup mogą być pobierane nawet wtedy, gdy grupa jest niekompletna.

GMAMSA oznacza GMASGA, który nie musi być określony.

Parametr GMAMSA można określić z dowolną z pozostałych opcji GM* oraz z dowolną z opcji MO*.

GMASGA,
Wszystkie segmenty w komunikacie logicznym muszą być dostępne.

Ta opcja określa, że segmenty w komunikacie logicznym stają się dostępne do pobrania tylko wtedy, gdy wszystkie segmenty w komunikacie logicznym są dostępne. Jeśli kolejka zawiera segmentowane komunikaty z brakującymi niektórymi segmentami (na przykład dlatego, że zostały opóźnione w sieci i jeszcze nie dotarły), podanie parametru GMASGA uniemożliwia pobranie segmentów należących do niekompletnych komunikatów logicznych. Jednak te segmenty nadal mają wpływ na wartość atrybutu kolejki **CurrentQDepth** . Oznacza to, że mogą nie istnieć żadne możliwe do pobrania komunikaty logiczne, nawet jeśli wartość **CurrentQDepth** jest większa od zera. Jeśli nie ma innych komunikatów, które można pobrać, kod przyczyny RC2033 jest zwracany po upływie określonego czasu oczekiwania (jeśli istnieje).

Przetwarzanie GMASGA zależy od tego, czy określono również GMLOGO:

- Jeśli obie opcje są określone, GMASGA działa tylko wtedy, gdy nie ma bieżącego komunikatu logicznego. Jeśli istnieje bieżący komunikat logiczny, GMASGA jest ignorowany. Oznacza to, że GMASGA może pozostać włączony podczas przetwarzania komunikatów w porządku logicznym.
- Jeśli GMASGA jest określony bez GMLOGO, GMASGA zawsze daje efekt. Oznacza to, że opcja musi zostać wyłączona po usunięciu z kolejki pierwszego segmentu komunikatu logicznego, aby można było usunąć pozostałe segmenty komunikatu logicznego.

Jeśli ta opcja nie zostanie podana, segmenty komunikatów mogą zostać pobrane nawet wtedy, gdy komunikat logiczny jest niekompletny.

Podczas gdy zarówno GMCMPM, jak i GMASGA wymagają, aby wszystkie segmenty były dostępne, zanim będzie można je pobrać, pierwszy z nich zwraca pełny komunikat, podczas gdy drugi z nich pozwala na pobranie segmentów jeden po drugim.

Jeśli dla komunikatu raportu określono GMASGA, menedżer kolejek wykonuje przetwarzanie specjalne. Menedżer kolejek sprawdza kolejkę, aby sprawdzić, czy dla każdego z segmentów, które tworzą pełny komunikat logiczny, istnieje co najmniej jeden komunikat raportu. Jeśli istnieje, warunek GMASGA jest spełniony. Jednak menedżer kolejek nie sprawdza typu komunikatów raportu, dlatego w komunikatach raportu dotyczących segmentów komunikatu logicznego może występować mieszanka typów raportów. W rezultacie sukces GMASGA nie oznacza, że GMCMPM odniesie sukces. Jeśli istnieje kombinacja typów raportów dla segmentów konkretnego komunikatu logicznego, te komunikaty muszą być pobierane pojedynczo.

GMASGA można podać z dowolną z pozostałych opcji GM* i z dowolną z opcji MO*.

Opcja domyślna: Jeśli żadna z opisanych opcji nie jest wymagana, można użyć następującej opcji:

BRAK GMNONE
Nie określono żadnych opcji.

Ta wartość może być używana do wskazania, że nie określono żadnych innych opcji; wszystkie opcje przyjmują wartości domyślne. GMNONE jest zdefiniowane w celu wspomagania dokumentacji programu; nie jest zamierzone, aby ta opcja była używana z innymi, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Wartością początkową pola *GMOPT* jest GMNWT.

GMRE1 (łańcuch znaków o długości 1 bajtu)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość początkowa tego pola jest znakiem odstępu. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER2.

GMRL (10-cyfrowa liczba całkowita ze znakiem)

Długość zwracanych danych komunikatu (w bajtach).

Jest to pole wyjściowe ustawiane przez menedżer kolejek na długość (w bajtach) danych komunikatu zwracanych przez wywołanie MQGET w parametrze **BUFFER** . Jeśli menedżer kolejek nie obsługuje tej możliwości, parametr *GMRL* jest ustawiany na wartość RLUNDF.

Podczas konwersji komunikatów między kodowaniami lub zestawami znaków dane komunikatu mogą czasami zmieniać wielkość. Przy powrocie z wywołania MQGET:

- Jeśli parametr *GMRL* nie ma wartości RLUNDF, *GMRL*zwraca liczbę bajtów danych komunikatu.
- Jeśli parametr *GMRL* ma wartość RLUNDF, liczba bajtów zwracanych danych komunikatu jest zwykle podawana przez mniejszą z wartości *BUFLEN* i *DATLEN*, ale może być mniejsza, jeśli wywołanie MQGET zakończy się z kodem przyczyny RC2079 . W takiej sytuacji nieistotne bajty w parametrze **BUFFER** są ustawiane na wartości null.

Zdefiniowana jest następująca wartość specjalna:

RLUNDF
Nie zdefiniowano długości zwróconych danych.

Wartością początkową tego pola jest RLUNDF. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER3.

GMRQN (48-bajtowy łańcuch znaków)

Przetłumaczona nazwa kolejki docelowej.

Jest to pole wyjściowe, które jest ustawiane przez menedżer kolejek na nazwę lokalną kolejki, z której został pobrany komunikat, zgodnie z definicją w lokalnym menedżerze kolejek. Różni się to od nazwy użytej do otwarcia kolejki, jeśli:

- Kolejka aliasowa została otwarta (w takim przypadku zwracana jest nazwa kolejki lokalnej, do której alias został przetłumaczony), lub
- Otwarto kolejkę modelową (w tym przypadku zwracana jest nazwa dynamicznej kolejki lokalnej).

Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

GMRS2 (1-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone. Wartość początkowa tego pola jest znakiem odstępu. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER4.

GMSEG (łańcuch znaków o długości 1 bajtu)

Flaga wskazująca, czy dla pobieranego komunikatu dozwolona jest dalsza segmentacja.

Ma jedną z następujących wartości:

SEGIHB
Segmentacja niedozwolona.

SEGALWunit description in lists
Segmentacja jest dozwolona.

Jest to pole wyjściowe. Wartością początkową tego pola jest SEGIHB. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER2.

GMSG1 (10-cyfrowa liczba całkowita ze znakiem)

Sygnał.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

GMSG2 (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator sygnału.

Jest to pole zastrzeżone; jego wartość nie jest istotna.

GMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

GMSIDV
Identyfikator struktury opcji pobierania komunikatu.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest GMSIDV.

GMSST (łańcuch znaków jednobajtowych)

Flaga wskazująca, czy pobrany komunikat jest segmentem komunikatu logicznego.

Ma jedną z następujących wartości:

SSNSEG,
Komunikat nie jest segmentem.

SSSEG
Komunikat jest segmentem, ale nie jest ostatnim segmentem komunikatu logicznego.

SSLSEG,
Komunikat jest ostatnim segmentem komunikatu logicznego.

Jest to również wartość zwracana, jeśli komunikat logiczny składa się tylko z jednego segmentu.

To pole jest polem wyjściowym. Wartością początkową tego pola jest SSNSEG. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER2.

GMTOK (16-bajtowy łańcuch bitowy)

Znacznik komunikatu.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Zdefiniowana jest następująca wartość specjalna:

MMTKNON
Brak znacznika komunikatu.

Wartością długości pola jest zero binarne.

Długość tego pola jest określona przez LNMTOK. Wartością początkową tego pola jest MTKNON. To pole jest ignorowane, jeśli wartość *GMVER* jest mniejsza niż GMVER3.

GMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

GMVER1
Version-1 -struktura opcji get-message.

GMVER2
Version-2 -struktura opcji get-message.

GMVER3
Version-3 -struktura opcji get-message.

GMVER4
Version-4 -struktura opcji get-message.

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

GMVERC
Bieżąca wersja struktury opcji pobierania komunikatów.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest GMVER1.

GMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

GMVER1
Version-1 -struktura opcji get-message.

GMVER2
Version-2 -struktura opcji get-message.

GMVER3
Version-3 -struktura opcji get-message.

GMVER4
Version-4 -struktura opcji get-message.

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

GMVERC
Bieżąca wersja struktury opcji pobierania komunikatów.

To pole jest zawsze polem wejściowym. Wartością początkową tego pola jest GMVER1.

GMWI (10-cyfrowa liczba całkowita ze znakiem)

Interwał oczekiwania.

Jest to przybliżony czas (wyrażony w milisekundach), przez który wywołanie MQGET oczekuje na nadejście odpowiedniego komunikatu (to znaczy komunikatu spełniającego kryteria wyboru określone

w parametrze **MSGDSC** wywołania MQGET; więcej szczegółów zawiera pole *MDMID* opisane w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141). Jeśli po upływie tego czasu nie zostanie odebrany odpowiedni komunikat, wywołanie zostanie zakończone z kodem CCFAIL i kodem przyczyny RC2033.

Parametr *GMWI* jest używany z opcją GMWT. Jeśli ta opcja nie zostanie podana, zostanie ona zignorowana. Jeśli zostanie podana, wartość *GMWI* musi być większa lub równa zero lub mieć następującą wartość specjalną:

WIULIM
Nieograniczony czas oczekiwania.

Wartością początkową tego pola jest 0.

Wartości początkowe

Tabela 704. Pola w MQGMO

Nazwa pola	Nazwa stałej	Wartość stałej
GMSID	GMSIDV	'GMO¬'
GMVER	GMVER1	1
GMOPT	GMNWT,	0
GMWI	Brak	0
GMSG1	Brak	0
GMSG2	Brak	0
GMRQN	Brak	Puste
GMMO	MOMSGI + MOCORI	3
GMGST	GSNIG@ item: indo	''
GMSST	SSNSEG,	''
GMSEG	SEGIHB	''
GMRE1	Brak	''
GMTOK	MMTKNON	Wartości null
GMRL	RLUNDF	-1
GMRS2	Brak	''
GMMH	BRAK HMC	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQGMO Structure
D*
D* Structure identifier
D  GMSID                  1      4    INZ('GMO ')
D* Structure version number
D  GMVER                  5      8I 0 INZ(1)
D* Options that control the action ofMQGET
D  GMOPT                  9     12I 0 INZ(0)
D* Wait interval
```

```
D  GMWI                  13     16I 0 INZ(0)
D* Signal
D  GMSG1                 17     20I 0 INZ(0)
D* Signal identifier
D  GMSG2                 21     24I 0 INZ(0)
D* Resolved name of destination queue
D  GMRQN                 25     72    INZ
D* Options controlling selection criteriaused for MQGET
D  GMMO                  73     76I 0 INZ(3)
D* Flag indicating whether messageretrieved is in a group
D  GMGST                 77     77    INZ(' ')
D* Flag indicating whether messageretrieved is a segment of a
D* logicalmessage
D  GMSST                 78     78    INZ(' ')
D* Flag indicating whether furthersegmentation is allowed for themessage
D* retrieved
D  GMSEG                 79     79    INZ(' ')
D* Reserved
D  GMRE1                 80     80    INZ
D* Message token
D  GMTOK                 81     96    INZ(X'00000000000000-
D                                     000000000000000000')
D* Length of message data returned(bytes)
D  GMRL                  97    100I 0 INZ(-1)
D* Reserved
D  GMRS2                101    104I 0 INZ(0)
D* Message handle
D  GMMH                 105    112I 0 INZ(0)
```

IBM i MQIIH (nagłówek informacjiIMS) w systemie IBM i

Struktura MQIIH opisuje informacje, które muszą być dostępne na początku komunikatu wysyłanego do mostu IMS przez IBM MQ for z/OS.

Przegląd

Nazwa formatu: FMIMS.

Zestaw znaków i kodowanie: specjalne warunki mają zastosowanie do zestawu znaków i kodowania używanego dla struktury MQIIH i danych komunikatu aplikacji:

- Aplikacje łączące się z menedżerem kolejek, który jest właścicielem kolejki mostu IMS , muszą udostępniać strukturę MQIIH, która jest w zestawie znaków i kodowaniu menedżera kolejek. Jest to spowodowane tym, że w tym przypadku nie jest wykonywana konwersja danych struktury MQIIH.
- Aplikacje łączące się z innymi menedżerami kolejek mogą udostępnić strukturę MQIIH, która znajduje się w dowolnym z obsługiwanych zestawów znaków i kodowań. Konwersja MQIIH jest wykonywana przez odbierającego agenta kanału komunikatów połączonego z menedżerem kolejek, który jest właścicielem kolejki mostu IMS .

  Uwaga: Istnieje jeden wyjątek. Jeśli menedżer kolejek, który jest właścicielem kolejki mostu IMS , używa produktu CICS na potrzeby kolejkowania rozproszonego, produkt MQIIH musi być w zestawie znaków i kodowaniu menedżera kolejek, który jest właścicielem kolejki mostu IMS .
- Dane komunikatu aplikacji po strukturze MQIIH muszą mieć ten sam zestaw znaków i kodowanie, co struktura MQIIH. Pola *IICSI* i *IIENC* w strukturze MQIIH nie mogą być używane do określania zestawu znaków i kodowania danych komunikatu aplikacji.

  Jeśli dane nie są jednym z wbudowanych formatów obsługiwanych przez menedżer kolejek, użytkownik musi podać wyjście konwersji danych, aby dokonać konwersji danych komunikatu aplikacji.

- “Uwierzytelnianie przepustek dla aplikacji mostu IMS” na stronie 1130
- “Pola” na stronie 1130
- “Wartości początkowe” na stronie 1133
- “Deklaracja RPG” na stronie 1133

Uwierzytelnianie przepustek dla aplikacji mostu IMS

Administratorzy systemu IBM MQ mogą teraz określić nazwę aplikacji, która ma być używana do uwierzytelniania przepustek dla aplikacji mostu IMS . W tym celu nazwa aplikacji jest określona jako nowy atrybut PTKTAPPL dla definicji obiektu STGCLASS, jako łańcuch alfanumeryczny o długości od 1 do 8 znaków.

Pusta wartość oznacza, że uwierzytelnianie odbywa się tak jak w poprzednich wersjach produktu IBM MQ, to znaczy, że w żądaniu uwierzytelniania nie przepływa żadna nazwa aplikacji, a zamiast tego używana jest wartość MVSxxxx.

Wartość 1-8 znaków alfanumerycznych musi być zgodna z regułami dotyczącymi nazw aplikacji przepustek opisanymi w publikacjach RACF .

Zarówno administratorzy IBM MQ , jak i administratorzy RACF muszą zgadzać się na poprawne nazwy aplikacji, które mają być używane. Administrator RACF musi utworzyć profil w klasie PTKTDATA przyznający dostęp do odczytu do identyfikatorów użytkowników wszystkich aplikacji, którym ma zostać nadany dostęp. Administrator IBM MQ musi utworzyć lub zmienić wymagane definicje STGCLASS określające nazwę aplikacji, która ma być używana do uwierzytelniania za pomocą przepustki.

Informacje pokrewne zawiera publikacja *Script (MQSC) Command Reference*.

Pola

Struktura MQIIH zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

IIAUT (8-bajtowy łańcuch znaków)

Hasło lub przepustka RACF .

Ten parametr jest opcjonalny. Jeśli zostanie określony, jest on używany z identyfikatorem użytkownika w kontekście zabezpieczeń MQMD do budowania znacznika UTOKEN, który jest wysyłany do programu IMS w celu udostępnienia kontekstu zabezpieczeń. Jeśli nie zostanie podany, ID użytkownika zostanie użyty bez weryfikacji. Zależy to od ustawienia przełączników RACF , które mogą wymagać obecności elementu uwierzytelniającego.

Ta opcja jest ignorowana, jeśli pierwszy bajt jest pusty lub ma wartość NULL. Można użyć następującej wartości specjalnej:

IAUNON
Brak uwierzytelniania.

Długość tego pola jest określona przez LNAUTH. Wartością początkową tego pola jest IAUNON.

IICMT (1-bajtowy łańcuch znaków)

Tryb zatwierdzania.

Więcej informacji na temat trybów zatwierdzania IMS zawiera publikacja *OTMA Reference* . Wartość musi być jedną z następujących wartości:

Tabela ICMCTS
Zatwierdź, a następnie wyślij.

Ten tryb oznacza podwójne kolejkowanie danych wyjściowych, ale krótsze czasy zajętości regionów. Transakcje szybkie i konwersacyjne nie mogą działać w tym trybie.

Tabela ICMSTC
Wyślij, a następnie zatwierdź.

Wartością początkową tego pola jest ICMCTS.

IICSI (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

IIENC (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest 0.

IIFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Wartość musi być następująca:

IINON
Brak flag.

Wartością początkową tego pola jest IINONE.

IIFMT (8-bajtowy łańcuch znaków)

Nazwa formatu IBM MQ danych następujących po MQIIH.

Określa nazwę formatu IBM MQ danych, które są zgodne ze strukturą MQIIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Długość tego pola jest określona przez LNFMT. Wartością początkową tego pola jest FMNONE.

IILEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQIIH.

Wartość musi być następująca:

IILEN1
Długość struktury nagłówka informacji IMS .

Wartością początkową tego pola jest IILEN1.

IILTO (8-bajtowy łańcuch znaków)

Przesłonięcie terminalu logicznego.

Jest on umieszczany w polu bloku PCB we/wy. Jest ona opcjonalna; jeśli nie jest określona, używana jest nazwa potoku TPIPE. Jest on ignorowany, jeśli pierwszy bajt jest pusty lub pusty.

Długość tego pola jest określona przez LNLTOV. Wartość początkowa tego pola to 8 znaków odstępu.

IIMMN (8-bajtowy łańcuch znaków)

Nazwa odwzorowania usług formatu komunikatu.

Jest on umieszczany w polu bloku PCB we/wy. Jest ono opcjonalne. Na wejściu reprezentuje MID, na wyjściu reprezentuje MOD. Jest on ignorowany, jeśli pierwszy bajt jest pusty lub pusty.

Długość tego pola jest określona przez LNMFMN. Wartość początkowa tego pola to 8 znaków odstępu.

IIRFM (8-bajtowy łańcuch znaków)

Nazwa formatu IBM MQ komunikatu odpowiedzi.

Jest to nazwa formatu IBM MQ komunikatu odpowiedzi, który zostanie wysłany w odpowiedzi na bieżący komunikat. Reguły kodowania są takie same, jak w przypadku pola *MDFMT* w strukturze MQMD.

Długość tego pola jest określona przez LNFMT. Wartością początkową tego pola jest FMNONE.

IIRSV (1-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone; musi być puste.

IISEC (1-bajtowy łańcuch znaków)

Zasięg zabezpieczeń.

Wskazuje to wymagane przetwarzanie zabezpieczeń systemu IMS . Zdefiniowane są następujące wartości:

ISSCHK,
Sprawdź zasięg zabezpieczeń.

Platforma ACEE jest zbudowana w regionie sterującym, ale nie w regionie zależnym.

SSFUL
Pełny zasięg zabezpieczeń.

Buforowana platforma ACEE jest budowana w regionie sterowania, a niebuforowana platforma ACEE jest budowana w regionie zależnym. Jeśli używany jest interfejs ISSFUL, należy upewnić się, że identyfikator użytkownika, dla którego zbudowano ACEE, ma dostęp do zasobów używanych w regionie zależnym.

Jeśli ISSCHK i ISSFUL nie są określone dla tego pola, przyjmowany jest ISSCHK.

Wartością początkową tego pola jest ISSCHK.

IISID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

IISIDV
Identyfikator struktury nagłówka informacji IMS .

Wartością początkową tego pola jest IISIDV.

IITID (16-bajtowy łańcuch bitowy)

Identyfikator instancji transakcji.

To pole jest używane przez komunikaty wyjściowe z produktu IMS , dlatego jest ignorowane w przypadku pierwszego wejścia. Jeśli parametr *IITST* jest ustawiony na wartość ITSIC, należy go podać w następnym wejściu i we wszystkich kolejnych wejściach, aby umożliwić produktowi IMS korelowanie komunikatów z poprawną konwersacją. Można użyć następującej wartości specjalnej:

ITINON
Brak identyfikatora instancji transakcji.

Długość tego pola jest określona przez LNTIID. Wartością początkową tego pola jest ITINON.

IITST (1-bajtowy łańcuch znaków)

Stan transakcji.

Wskazuje stan konwersacji IMS . Jest ona ignorowana przy pierwszym wejściu, ponieważ nie istnieje żadna konwersacja. Przy kolejnych danych wejściowych wskazuje, czy konwersacja jest aktywna, czy nie. Na wyjściu jest on ustawiany przez IMS. Wartość musi być jedną z następujących wartości:

ITSIC,
W konwersacji.

ITSNINA
Nie w konwersacji.

ITSARC
Zwraca dane stanu transakcji w postaci strukturalnej.

Ta wartość jest używana tylko z komendą IMS `/DISPLAY TRAN` . Powoduje, że dane stanu transakcji są zwracane w postaci IMS , a nie w postaci znakowej. Więcej informacji na ten temat zawiera sekcja Tworzenie programów transakcyjnych IMS za pośrednictwem produktu IBM MQ .

Wartością początkową tego pola jest ITSNIC.

IIVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

IIVER1
Numer wersji struktury nagłówka informacji IMS .

Następująca stała określa numer wersji bieżącej:

IIVERC
Bieżąca wersja struktury nagłówka informacji IMS .

Wartością początkową tego pola jest IIVER1.

Wartości początkowe

Tabela 705. Pola w MQIIH

Nazwa pola	Nazwa stałej	Wartość stałej
IISID	IISIDV	'IIH¬'
IIVER	IIVER1	1
IILEN	IILEN1	84
IIENC	Brak	0
IICSI	Brak	0
IIFMT	BRAK FMNONE	Puste
IIFLG	IINON	0
IILTO	Brak	Puste
IIMMN	Brak	Puste
IIRFM	BRAK FMNONE	Puste
IIAUT	IAUNON	Puste
IITID	ITINON	Wartości null
IITST	ITSNINA	''
IICMT	Tabela ICMCTS	'0'
IISEC	ISSCHK,	'C'
IIRSV	Brak	''

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQIIH Structure
D*
D* Structure identifier
D  IISID                 1      4    INZ('IIH ')
D* Structure version number
D  IIVER                 5      8I 0 INZ(1)
D* Length of MQIIH structure
D  IILEN                 9     12I 0 INZ(84)
```

```
     D* Reserved
     D  IIENC                 13     16I 0 INZ(0)
     D* Reserved
     D  IICSI                 17     20I 0 INZ(0)
     D* MQ format name of data that followsMQIIH
     D  IIFMT                 21     28    INZ('        ')
     D* Flags
     D  IIFLG                 29     32I 0 INZ(0)
     D* Logical terminal override
     D  IILTO                 33     40    INZ
     D* Message format services map name
     D  IIMMN                 41     48    INZ
     D* MQ format name of reply message
     D  IIRFM                 49     56    INZ('        ')
     D* RACF password or passticket
     D  IIAUT                 57     64    INZ('        ')
     D* Transaction instance identifier
     D  IITID                 65     80    INZ(X'00000000000000-
     D                                     000000000000000000')
     D* Transaction state
     D  IITST                 81     81    INZ(' ')
     D* Commit mode
     D  IICMT                 82     82    INZ('0')
     D* Security scope
     D  IISEC                 83     83    INZ('C')
     D* Reserved
     D  IIRSV                 84     84    INZ
```

IBM i MQIMPO (opcje właściwości komunikatu zapytania) w systemie IBM i

Struktura MQIMPO umożliwia aplikacjom określanie opcji sterujących sposobem uzyskiwania informacji o właściwościach komunikatów.

Przegląd

Cel: Struktura jest parametrem wejściowym wywołania MQINQMP.

Zestaw znaków i kodowanie: Dane w MQIMPO muszą być w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- “Pola” na stronie 1134
- “Wartości początkowe” na stronie 1140
- “Deklaracja RPG” na stronie 1140

Pola

Struktura MQIMPO zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

IPOPT (10-cyfrowa liczba całkowita ze znakiem)

Następujące opcje sterują działaniem MQINQMP. Można określić jedną lub więcej z tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe). Kombinacje opcji, które nie są poprawne, są odnotowywane; wszystkie pozostałe kombinacje są poprawne.

Opcje danych wartości: Następujące opcje odnoszą się do przetwarzania danych wartości po pobraniu właściwości z komunikatu.

IPCVAL
Ta opcja żąda, aby wartość właściwości została przekształcona w wartość zgodną z wartościami *IPREQCSI* i *IPREQENC* określonymi przed wywołaniem MQINQMP, która zwraca wartość właściwości w obszarze *Value* .

- Jeśli konwersja zakończy się pomyślnie, pola *IPRETCSI* i *IPRETENC* zostaną ustawione na wartość taką samą, jak pola *IPREQCSI* i *IPREQENC* po powrocie z wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wartość właściwości zostanie zwrócona bez konwersji.

  Jeśli właściwość jest łańcuchem, pola *IPRETCSI* i *IPRETENC* są ustawione na zestaw znaków i kodowanie nieprzekształconego łańcucha.

  W tym przypadku kod zakończenia to CCWARN, z kodem przyczyny RC2466. Kursor właściwości jest przenoszony do zwróconej właściwości.

Jeśli wartość właściwości jest rozszerzana podczas konwersji i przekracza wielkość parametru **Value** , wartość jest zwracana bez konwersji z kodem zakończenia CCFAIL; kod przyczyny jest ustawiany na RC2469.

Parametr **DataLength** wywołania MQINQMP zwraca długość, na jaką wartość właściwości zostałaby przekształcona, aby umożliwić aplikacji określenie wielkości buforu wymaganego do dostosowania przekształconej wartości właściwości. Kursor właściwości pozostaje niezmieniony.

Ta opcja wymaga również, aby:

- Jeśli nazwa właściwości zawiera znak wieloznaczny,
- Pole *IPRETNAMECHRP* jest inicjowane z adresem lub przesunięciem dla zwróconej nazwy,

wówczas zwracana nazwa jest przekształcana w celu zapewnienia zgodności z wartościami *IPREQCSI* i *IPREQENC* .

- Jeśli konwersja zakończy się pomyślnie, pole *VSCCSID* w pliku *IPRETNAMECHRP* i kodowanie zwróconej nazwy zostaną ustawione na wartość wejściową *IPREQCSI* i *IPREQENC*.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu lub ostrzeżenia, zwrócona nazwa nie zostanie przekształcona. W tym przypadku kod zakończenia to CCWARN, z kodem przyczyny RC2492.

  Kursor właściwości jest przenoszony do zwróconej właściwości. Jeśli wartość i nazwa nie zostaną przekształcone, zwracana jest wartość RC2466 .

Jeśli zwrócona nazwa zostanie rozszerzona podczas konwersji i zostanie przekroczona wielkość pola *VSBufsize* w pliku *RequestedName*, zwrócony łańcuch pozostanie nieprzekształcony z kodem zakończenia CCFAIL, a kod przyczyny zostanie ustawiony na RC2465.

Pole *VSLength* w strukturze MQCHARV zwraca długość, na jaką wartość właściwości zostałaby przekształcona, aby umożliwić aplikacji określenie wielkości buforu wymaganego do dostosowania przekształconej wartości właściwości. Kursor właściwości pozostaje niezmieniony.

IPCTYP

Ta opcja wymaga, aby wartość właściwości została przekształcona z bieżącego typu danych w typ danych określony w parametrze **Type** wywołania MQINQMP.

- Jeśli konwersja powiedzie się, parametr **Type** pozostanie niezmieniony po zwróceniu wywołania MQINQMP.
- Jeśli konwersja nie powiedzie się, ale wywołanie MQINQMP zakończy się bez błędu, wywołanie nie powiedzie się z kodem przyczyny RC2470. Kursor właściwości pozostaje niezmieniony.

Jeśli konwersja typu danych powoduje rozszerzenie wartości podczas konwersji, a przekształcona wartość przekracza wielkość parametru **Value** , zwracana jest wartość nieprzekształcona, z kodem zakończenia CCFAIL i kodem przyczyny RC2469.

Parametr **DataLength** wywołania MQINQMP zwraca długość, na jaką wartość właściwości zostałaby przekształcona, aby umożliwić aplikacji określenie wielkości buforu wymaganego do dostosowania przekształconej wartości właściwości. Kursor właściwości pozostaje niezmieniony.

Jeśli wartość parametru **Type** wywołania MQINQMP jest niepoprawna, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2473.

Jeśli żądana konwersja typu danych nie jest obsługiwana, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2470. Obsługiwane są następujące konwersje typów danych:

Tabela 706. Obsługiwane konwersje typów danych

Typ danych właściwości	Obsługiwane docelowe typy danych
TYPBOL	TYPSTR, TYPI8, TYPI16, TYPI32, TYPI64
TYPBST	TYPSTR
TYPI8	TYPSTR, TYPI16, TYPI32, TYPI64
TYPI16	TYPSTR, TYPI32, TYPI64
TYPI32	TYPSTR, TYPI64
TYPI64	TYPSTR
TYPF32	TYPSTR, TYPF64
TYPF64	TYPSTR
TYPSTR	TYPBOL, TYPI8, TYPI16, TYPI32, TYPI64, TYPF32, TYPF64
TYPNUL	Brak

Ogólne zasady dotyczące obsługiwanych konwersji są następujące:

- Wartości liczbowe właściwości mogą być przekształcane z jednego typu danych na inny, pod warunkiem, że podczas konwersji nie zostaną utracone żadne dane.

  Na przykład wartość właściwości o typie danych TYPI32 może zostać przekształcona w wartość o typie danych TYPI64, ale nie może zostać przekształcona w wartość o typie danych TYPI16.
- Wartość właściwości dowolnego typu danych można przekształcić w łańcuch.
- Wartość właściwości łańcuchowej może zostać przekształcona w dowolny inny typ danych, pod warunkiem, że łańcuch jest poprawnie sformatowany na potrzeby konwersji. Jeśli aplikacja podejmie próbę przekształcenia wartości właściwości łańcuchowej, która nie jest poprawnie sformatowana, program IBM MQ zwróci kod przyczyny RC2472.
- Jeśli aplikacja podejmie próbę konwersji, która nie jest obsługiwana, IBM MQ zwróci kod przyczyny RC2470.

Konkretne reguły przekształcania wartości właściwości z jednego typu danych na inny są następujące:

- Podczas przekształcania wartości właściwości TYPBOL w łańcuch, wartość TRUE jest przekształcana w łańcuch "TRUE", a wartość false jest przekształcana w łańcuch "FALSE".
- Podczas konwersji wartości właściwości TYPBOL na liczbowy typ danych, wartość TRUE jest przekształcana na jeden, a wartość FALSE jest przekształcana na zero.
- Podczas konwersji wartości właściwości łańcuchowej na wartość TYPBOL łańcuch "TRUE" lub "1" jest przekształcany na TRUE, a łańcuch "FALSE" lub "0" jest przekształcany na FALSE.

  Należy zauważyć, że w terminach "TRUE" i "FALSE" nie jest rozróżniana wielkość liter.

  Żaden inny łańcuch nie może zostać przekształcony; IBM MQ zwraca kod przyczyny RC2472.
- Podczas przekształcania wartości właściwości łańcuchowej w wartość o typie danych TYPI8, TYPI16, TYPI32 lub TYPI64łańcuch musi mieć następujący format:

```
[blanks][sign]digits
```

Znaczenia komponentów łańcucha są następujące:

blanks
Opcjonalne początkowe znaki odstępu

sign
Opcjonalny znak plus (+) lub minus (-).

digits
Ciągła sekwencja cyfr (0-9). Musi występować co najmniej jedna cyfra.

Po sekwencji cyfr łańcuch może zawierać inne znaki, które nie są cyframi, ale konwersja kończy się natychmiast po osiągnięciu pierwszego z tych znaków. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę całkowitą.

IBM MQ zwraca kod przyczyny RC2472 , jeśli łańcuch nie jest poprawnie sformatowany.

- Podczas przekształcania wartości właściwości łańcuchowej w wartość o typie danych TYPF32 lub TYPF64łańcuch musi mieć następujący format:

```
[blanks][sign]digits[.digits][e_char[e_sign]e_digits]
```

Znaczenia komponentów łańcucha są następujące:

blanks
Opcjonalne początkowe znaki odstępu

sign
Opcjonalny znak plus (+) lub minus (-).

digits
Ciągła sekwencja cyfr (0-9). Musi występować co najmniej jedna cyfra.

e_char
Znak wykładnika, którym jest "E" lub "e".

e_sign
Opcjonalny znak plus (+) lub minus (-) dla wykładnika.

e_digits
Ciągła sekwencja cyfr (0-9) dla wykładnika. Jeśli łańcuch zawiera znak wykładnika, musi być obecny co najmniej jeden znak cyfry.

Po sekwencji cyfr lub opcjonalnych znaków reprezentujących wykładnik łańcuch może zawierać inne znaki, które nie są znakami cyfrowymi, ale konwersja kończy się natychmiast po osiągnięciu pierwszego z tych znaków. Przyjmuje się, że łańcuch reprezentuje dziesiętną liczbę zmiennopozycyjną z wykładnikiem, który jest potęgą liczby 10.

IBM MQ zwraca kod przyczyny RC2472 , jeśli łańcuch nie jest poprawnie sformatowany.

- Podczas przekształcania liczbowej wartości właściwości w łańcuch, wartość jest przekształcana w łańcuchową reprezentację wartości jako liczba dziesiętna, a nie w łańcuch zawierający znak ASCII dla tej wartości. Na przykład liczba całkowita 65 jest przekształcana w łańcuch "65", a nie w łańcuch "A".
- Podczas konwersji wartości właściwości łańcucha bajtowego na łańcuch, każdy bajt jest przekształcany w dwa znaki szesnastkowe reprezentujące ten bajt. Na przykład tablica bajtów {0xF1, 0x12, 0x00, 0xFF} jest przekształcana w łańcuch "F11200FF".

IPQLEN
Zapytanie o typ i długość wartości właściwości. Długość jest zwracana w parametrze **DataLength** wywołania MQINQMP. Wartość właściwości nie jest zwracana.

Jeśli określono bufor *ReturnedName* , pole *VSLength* struktury MQCHARV jest wypełniane długością nazwy właściwości. Nazwa właściwości nie jest zwracana.

Opcje iteracji: Następujące opcje odnoszą się do iteracji właściwości przy użyciu nazwy ze znakiem wieloznacznym

IPINQF
Sprawdź pierwszą właściwość, która jest zgodna z podaną nazwą. Po tym wywołaniu dla zwracanej właściwości jest ustanawiany kursor.

Jest to wartość domyślna.

Opcja IPINQC może być następnie użyta z wywołaniem MQINQMP, jeśli jest to wymagane, w celu ponownego sprawdzenia tej samej właściwości.

Należy zauważyć, że istnieje tylko jeden kursor właściwości. Oznacza to, że jeśli nazwa właściwości określona w wywołaniu MQINQMP zostanie zmieniona, kursor zostanie zresetowany.

Ta opcja nie jest poprawna z żadną z następujących opcji:

IPINQN
IPINQC

IPINQN
Pyta o następną właściwość, która jest zgodna z podaną nazwą, kontynuując wyszukiwanie od kursora właściwości. Kursor jest przenoszony do zwracanej właściwości.

Jeśli jest to pierwsze wywołanie MQINQMP dla podanej nazwy, zwracana jest pierwsza właściwość zgodna z podaną nazwą.

Opcja IPINQC może być następnie użyta z wywołaniem MQINQMP, jeśli jest to wymagane, w celu ponownego sprawdzenia tej samej właściwości.

Jeśli właściwość pod kursorem została usunięta, MQINQMP zwraca następną zgodną właściwość po tej, która została usunięta.

Jeśli zostanie dodana właściwość, która jest zgodna ze znakiem wieloznacznym, podczas gdy iteracja jest w toku, właściwość może, ale nie musi, zostać zwrócona podczas kończenia iteracji. Ta właściwość jest zwracana po zrestartowaniu iteracji za pomocą IPINQF.

Właściwość zgodna ze znakiem wieloznacznym, który został usunięty, gdy iteracja była w toku, nie jest zwracana po jej usunięciu.

Ta opcja nie jest poprawna z żadną z następujących opcji:

IPINQF
IPINQC

IPINQC
Pobierz wartość właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości jest tą, która została ostatnio sprawdzona przy użyciu opcji IPINQF lub IPINQN.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany, gdy uchwyt komunikatu jest określony w polu *MsgHandle* obiektu MQGMO w wywołaniu MQGET lub gdy uchwyt komunikatu jest określony w polach *OriginalMsgHandle* lub *NewMsgHandle* struktury MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze ustawiony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i przyczyną RC2471.

Ta opcja nie jest poprawna z żadną z następujących opcji:

IPINQF
IPINQN

Jeśli żadna z opisanych wcześniej opcji nie jest wymagana, można użyć następującej opcji:

BRAK IP
Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

IPNONE jest pomocne w dokumentacji programu; opcja ta nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, użycie tej opcji nie może zostać wykryte.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest IPINQF.

IPREQCSI (10-cyfrowa liczba całkowita ze znakiem)

Zestaw znaków, na który ma zostać przekształcona wartość właściwości zapytania, jeśli wartość jest łańcuchem znaków. Jest to także zestaw znaków, na który ma być konwertowany *ReturnedName* , jeśli określono IPCVAL lub IPCTYP.

Wartością początkową tego pola jest CSAPL.

IPREQENC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kodowanie, na które ma zostać przekształcona wartość właściwości zapytania, jeśli określono parametr IPCVAL lub IPCTYP.

Wartością początkową tego pola jest ENNAT.

IPRE1 (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole zastrzeżone. Wartość początkowa tego pola jest znakiem odstępu.

IPRETCSI (10-cyfrowa liczba całkowita ze znakiem)

W danych wyjściowych jest to zestaw znaków wartości zwracanej, jeśli parametr **Type** wywołania MQINQMP ma wartość TYPSTR.

Jeśli podano opcję IPCVAL i konwersja zakończyła się pomyślnie, pole *ReturnedCCSID* w przypadku powrotu ma taką samą wartość, jak przekazana wartość.

Wartością początkową tego pola jest zero.

IPRETENC (10-cyfrowa liczba całkowita ze znakiem)

Na wyjściu jest to kodowanie zwróconej wartości.

Jeśli podano opcję IPCVAL i konwersja zakończyła się pomyślnie, pole *ReturnedEncoding* w przypadku powrotu ma taką samą wartość, jak przekazana wartość.

Wartością początkową tego pola jest ENNAT.

IPRETNAMCHRP (10-cyfrowa liczba całkowita ze znakiem)

Rzeczywista nazwa właściwości, do której wysłano zapytanie.

W przypadku danych wejściowych bufor łańcucha można przekazać przy użyciu pola *VSPtr* lub *VSOffset* struktury MQCHARV. Długość buforu łańcucha jest określana za pomocą pola *VSBufsize* struktury MQCHARV.

Po powrocie z wywołania MQINQMP bufor łańcucha jest uzupełniany nazwą właściwości, której dotyczy zapytanie, pod warunkiem, że bufor łańcucha był wystarczająco długi, aby w pełni zawierał nazwę. Pole *VSLength* struktury MQCHARV jest wypełniane długością nazwy właściwości. Pole *VSCCSID* struktury MQCHARV jest wypełniane w celu wskazania zestawu znaków zwracanej nazwy, niezależnie od tego, czy konwersja nazwy nie powiodła się.

Jest to pole wejściowe/wyjściowe. Wartością początkową tego pola jest MQCHARV_DEFAULT.

IPSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury. Wartość musi być następująca:

IPSIDV
Identyfikator struktury opcji właściwości komunikatu zapytania.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest IPSIDV.

IPTYP (10-cyfrowa liczba całkowita ze znakiem)

Reprezentacja łańcuchowa typu danych właściwości.

Jeśli właściwość została określona w nagłówku MQRFH2 i atrybut MQRFH2 `dt` nie został rozpoznany, można użyć tego pola do określenia typu danych właściwości. Kod *TypeString* jest zwracany w kodowanym zestawie znaków 1208 (UTF-8) i jest to pierwsze osiem bajtów wartości atrybutu `dt` właściwości, której rozpoznawanie nie powiodło się.

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest łańcuch pusty w języku programowania C i 8 znaków odstępu w innych językach programowania.

IPVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

IPVER1
Numer wersji struktury opcji właściwości komunikatu zapytania.

Następująca stała określa numer wersji bieżącej:

IPVERC
Bieżąca wersja struktury opcji właściwości komunikatu zapytania.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest IPVER1.

Wartości początkowe

Tabela 707. Pola w programie MQIPMO

Nazwa pola	Nazwa stałej	Wartość stałej
IPSID	IPSIDV	'IMPO'
IPVER	IPVER1	1
IPOPT	IPINQF	
IPREQENC	ENNAT,	
IPREQCSI	CSAPL	
IPRETENC	ENNAT,	
IPRETCSI	0	
IPRE1	0	
IPRETNAMCHRP		
IPTYP		wartości puste

Deklaracja RPG

```
D* MQIMPO Structure
D*
D*
D* Structure identifier
D IPSID         1   4  INZ('IMPO')
D*
D* Structure version number
D IPVER         5   8I 0 INZ(1)
D*
** Options that control the action of
D* MQINQMP
D IPOPT         9   12I 0 INZ(0)
D*
D* Requested encoding of Value
```

```
D IPREQENC       13   16I 0 INZ(273)
D*
** Requested character set identifier
D* of Value
D IPREQCSI       17   20I 0 INZ(-3)
D*
D* Returned encoding of Value
D IPRETENC       21   24I 0 INZ(273)
D*
** Returned character set identifier of
D* Value
D IPRETCSI       25   28I 0 INZ(0)
D*
D* Reserved
D IPRE1         29   32I 0 INZ(0)
D*
D* Returned property name
D* Address of variable length string
D IPRETNAMCHRP     33   48*  INZ(*NULL)
D* Offset of variable length string
D IPRETNAMCHRO     49   52I 0 INZ(0)
D* Size of buffer
D IPRETNAMVSBS     53   56I 0 INZ(-1)
D* Length of variable length string
D IPRETNAMCHRL     57   60I 0 INZ(0)
D* CCSID of variable length string
D IPRETNAMCHRC     61   64I 0 INZ(-3)
D*
D* Property data type as a string
D IPTYP         65   72  INZ
```

IBM i MQMD (deskryptor komunikatu) w systemie IBM i

Przegląd

Cel: Struktura MQMD zawiera informacje sterujące, które towarzyszą danych aplikacji, gdy komunikat przemieszcza się między aplikacjami wysyłającymi i odbierającymi. Struktura jest parametrem wejścia/wyjścia w wywołaniach MQGET, MQPUT i MQPUT1 .

Wersja: Bieżąca wersja deskryptora MQMD to MDVER2. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Udostępniony plik COPY zawiera najnowszą wersję deskryptora MQMD obsługiwaną przez środowisko, ale z wartością początkową pola MDVER ustawioną na wartość MDVER1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu MDVER numer wersji wymaganej wersji.

Dostępna jest deklaracja struktury version-1 o nazwie MQMD1.

Zestaw znaków i kodowanie: dane w strukturze MQMD muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** oraz kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako IBM MQ MQI client, struktura musi być w zestawie znaków i kodowaniu klienta.

Jeśli nadawcze i odbiorcze menedżery kolejek używają różnych zestawów znaków lub kodowań, dane w strukturze MQMD są przekształcane automatycznie. Przekształcanie deskryptora MQMD przez aplikację nie jest konieczne.

- “Korzystanie z różnych wersji deskryptora MQMD” na stronie 1142
- “Kontekst komunikatu” na stronie 1142
- “Utrata ważności komunikatu” na stronie 1142
- “Pola” na stronie 1143
- “Wartości początkowe” na stronie 1185
- “Deklaracja RPG” na stronie 1186

Korzystanie z różnych wersji deskryptora MQMD

Struktura MQMD version-2 jest generalnie równoważna użyciu deskryptora MQMD version-1 i poprzedzaniu danych komunikatu strukturą MQMDE. Jeśli jednak wszystkie pola w strukturze MQMDE mają wartości domyślne, można pominąć MQMDE. W dalszej części tej sekcji opisano użycie deskryptora MQMD w wersji version-1 i deskryptora MQMDE.

- W wywołaniach MQPUT i MQPUT1 , jeśli aplikacja udostępnia deskryptor MQMD version-1 , aplikacja może opcjonalnie poprzedzić dane komunikatu przedrostkiem MQMDE, ustawiając pole MDFMT w deskryptorze MQMD na wartość FMMDE w celu wskazania, że istnieje MQMDE. Jeśli aplikacja nie udostępnia środowiska MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w środowisku MQMDE.

  Uwaga: Niektóre pola, które istnieją w strukturze MQMD version-2 , ale nie w strukturze MQMD version-1 , są polami wejściowymi/wyjściowymi w wywołaniach MQPUT i MQPUT1 . Jednak menedżer kolejek nie zwraca żadnych wartości w odpowiednich polach w MQMDE na wyjściu z wywołań MQPUT i MQPUT1 . Jeśli aplikacja wymaga tych wartości wyjściowych, musi użyć deskryptora MQMD version-2 .
- W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD version-1 , menedżer kolejek dodaje przedrostek do komunikatu zwróconego z MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość inną niż domyślna. Pole MDFMT w strukturze MQMD będzie miało wartość FMMDE wskazującą, że istnieje struktura MQMDE.

Wartości domyślne używane przez menedżer kolejek w polach MQMDE są takie same, jak wartości początkowe tych pól, patrz Tabela 709 na stronie 1185.

Jeśli komunikat znajduje się w kolejce transmisji, niektóre pola w strukturze MQMD mają ustawione określone wartości. Szczegółowe informacje na ten temat zawiera sekcja “MQXQH (nagłówek kolejki transmisji) w systemie IBM i” na stronie 1284 .

Kontekst komunikatu

Niektóre pola w strukturze MQMD zawierają kontekst komunikatu. Zwykle:

- *Kontekst tożsamości* odnosi się do aplikacji, która pierwotnie umieściła komunikat.
- *Kontekst pochodzenia* odnosi się do aplikacji, która ostatnio wstawiła komunikat.
- *Kontekst użytkownika* odnosi się do aplikacji, która pierwotnie umieściła komunikat.

Te dwie aplikacje mogą być tą samą aplikacją, ale mogą być także różnymi aplikacjami (na przykład, gdy komunikat jest przekazywany z jednej aplikacji do innej).

Chociaż tożsamość i kontekst źródłowy zwykle mają znaczenie opisane wcześniej, treść obu typów pól kontekstu w programie MQMD zależy w rzeczywistości od opcji PM*, które są określone podczas umieszczania komunikatu. W rezultacie kontekst tożsamości nie musi być związany z aplikacją, która pierwotnie umieściła komunikat, a kontekst pochodzenia nie musi być powiązany z aplikacją, która ostatnio umieściła komunikat-zależy to od projektu pakietu aplikacji.

Istnieje jedna klasa aplikacji, która nigdy nie modyfikuje kontekstu komunikatu, a mianowicie agent kanału komunikatów (MCA). Adaptery MCA, które odbierają komunikaty od menedżerów kolejek zdalnych, używają opcji kontekstu PMSETA w wywołaniu MQPUT lub MQPUT1 . Dzięki temu odbierający agent MCA może zachować dokładnie kontekst komunikatu, który był przesyłany wraz z komunikatem od wysyłającego agenta MCA. Jednak w rezultacie kontekst źródłowy nie odnosi się do aplikacji, która ostatnio umieściła komunikat (odbierający agent MCA), ale odnosi się do wcześniejszej aplikacji, która umieściła komunikat (prawdopodobnie samej aplikacji źródłowej).

Więcej informacji na ten temat zawiera sekcja Kontekst komunikatu.

Utrata ważności komunikatu

Komunikaty, które utraciły ważność w załadowanej kolejce (kolejce, która została otwarta), są automatycznie usuwane z kolejki w rozsądnym czasie po utracie ważności. Niektóre inne nowe funkcje tej wersji produktu IBM MQ mogą powodować, że załadowane kolejki będą skanowane rzadziej niż

w poprzedniej wersji produktu, jednak komunikaty, które utraciły ważność w załadowanych kolejkach, są zawsze usuwane w rozsądnym okresie ważności.

Pola

Struktura MQMD zawiera następujące pola; pola są opisane w porządku alfabetycznym:

MDACC (32-bitowy łańcuch)

Token rozliczania.

Jest to część *kontekstu tożsamości* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

MDACC umożliwia aplikacji spowodowanie odpowiedniego obciążenia pracą wykonaną w wyniku wysłania komunikatu. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza ich treści.

Gdy menedżer kolejek generuje te informacje, są one ustawiane w następujący sposób:

- Pierwszy bajt pola jest ustawiony na długość informacji rozliczeniowych obecnych w kolejnych bajtach. Długość ta jest z zakresu od 0 do 30 i jest przechowywana w pierwszym bajcie jako binarna liczba całkowita.
- Drugi i kolejne bajty (określone w polu długości) są ustawione na informacje rozliczeniowe odpowiednie dla środowiska.
  - z/OS W systemie z/OS informacje rozliczeniowe są ustawione na:
    - W przypadku zadania wsadowego z/OS informacje rozliczeniowe z karty JES JOB lub z instrukcji JES ACCT na karcie EXEC (separatory przecinków są zmieniane na X'FF '). W razie potrzeby informacje te są obcinane do 31 bajtów.
    - W przypadku TSO jest to numer konta użytkownika.
    - W przypadku systemu CICSjest to identyfikator jednostki pracy jednostki logicznej 6.2 (UEPUOWDS) (26 bajtów).
    - W systemie IMS: 8-znakowa nazwa PSB połączona z 16-znakowym tokenem odtwarzania IMS .
  - IBM i W systemie IBM iinformacje rozliczeniowe są ustawiane na kod rozliczeniowy zadania.
  - Linux AIX W systemie AIX and Linuxinformacje rozliczeniowe są ustawiane na liczbowy identyfikator użytkownika w postaci znaków ASCII.
  - Windows W systemie Windowsinformacje rozliczeniowe są ustawiane na identyfikator bezpieczeństwa systemu Windows NT (SID) w formacie skompresowanym. Identyfikator SID jednoznacznie identyfikuje identyfikator użytkownika zapisany w polu *MDUID* . Jeśli identyfikator SID jest przechowywany w polu *MDACC* , pomijane jest 6-bajtowe uprawnienie identyfikatora (znajdujące się w trzecim i kolejnych bajtach identyfikatora SID). Na przykład, jeśli identyfikator SID Windows NT ma długość 28 bajtów, w polu *MDACC* zostaną zapisane 22 bajty informacji o identyfikatorze SID.
- Ostatni bajt jest ustawiany na typ tokenu rozliczania, jedną z następujących wartości:

  ATTCIC,
  CICS Identyfikator LUOW.

  ATTDOS
  Domyślny token rozliczeniowy PC DOS.

  ATTWNT
  Identyfikator zabezpieczeń Windows .

  ATT400
  IBM i token rozliczania.

ATTUNX
AIX and Linux identyfikator liczbowy.

ATTUSR
Token rozliczania zdefiniowany przez użytkownika.

ZAK.
Nieznany typ tokenu rozliczania.

Typ tokenu rozliczania jest ustawiany na wartość jawną tylko w następujących środowiskach:

– 

AIX

– 

IBM i

– 

Windows

i dla IBM MQ MQI clients połączonych z tymi systemami.

W innych środowiskach typ tokenu rozliczania jest ustawiany na wartość ATTUNK. W tych środowiskach można użyć pola MDPAT do określenia typu odebranego tokenu rozliczania.

- Wszystkie pozostałe bajty są ustawione na zero binarne.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono PMSETI lub PMSETA. Jeśli nie określono ani PMSETI, ani PMSETA, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość MDACC , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru MDACC , która jest przechowywana razem z komunikatem, jeśli zostanie zachowany (więcej informacji na temat zachowanych publikacji zawiera opis PMRET w sekcji “MQPMO (opcje umieszczania komunikatów) w systemie IBM i” na stronie 1208), ale nie jest używana jako parametr MDACC , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ udostępnia on wartość przesłaniającą parametr MDACC we wszystkich wysłanych do nich publikacjach. Jeśli komunikat nie ma kontekstu, pole jest całkowicie binarne zero.

Jest to pole wyjściowe wywołania MQGET.

To pole nie podlega żadnej translacji na podstawie zestawu znaków menedżera kolejek-pole jest traktowane jako łańcuch bitów, a nie jako łańcuch znaków.

Menedżer kolejek nie wykonuje żadnych działań z informacjami w tym polu. Aplikacja musi interpretować informacje, jeśli chce je wykorzystać do celów księgowych.

W polu *MDACC* można użyć następującej wartości specjalnej:

ACNONE (brak)
Nie określono tokenu rozliczania.

Wartością długości pola jest zero binarne.

Długość tego pola jest określona przez LNACCT. Wartością początkową tego pola jest ACNONE.

MDAID (32-bajtowy łańcuch znaków)

Dane aplikacji odnoszące się do tożsamości.

Jest to część *kontekstu tożsamości* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

MDAID to informacje, które są definiowane przez pakiet aplikacji i mogą być używane do udostępniania dodatkowych informacji o komunikacie lub jego twórcy. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu. Gdy menedżer kolejek generuje te informacje, jest całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono PMSETI lub PMSETA. Jeśli występuje znak o kodzie zero, menedżer kolejek przekształca znak o kodzie zero i wszystkie następujące po nim znaki w znaki puste. Jeśli nie określono ani PMSETI, ani PMSETA, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość MDAID , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru MDAID , która jest przechowywana razem z komunikatem, jeśli zostanie zachowana (więcej informacji na temat zachowanych publikacji zawiera opis PMRET), ale nie jest używana jako parametr MDAID , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ we wszystkich wysłanych do nich publikacjach podano wartość przesłaniającą parametr MDAID . Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez LNAIDD. Wartość początkowa tego pola to 32 znaki odstępu.

MDAOD (4-bajtowy łańcuch znaków)

Dane aplikacji dotyczące pochodzenia.

Jest to część *kontekstu źródłowego* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

MDAOD to informacje zdefiniowane przez pakiet aplikacji, których można użyć do udostępnienia dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiana przez aplikacje działające z odpowiednimi uprawnieniami użytkownika w celu wskazania, czy dane tożsamości są zaufane.

Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu. Gdy menedżer kolejek generuje te informacje, jest całkowicie pusty.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono wartość PMSETA. Wszelkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Znak o kodzie zero i wszystkie następujące po nim znaki są przekształcane przez menedżera kolejek w odstępy. Jeśli parametr PMSETA nie jest określony, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość MDAOD , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru MDAOD , która jest przechowywana razem z komunikatem, jeśli zostanie zachowana (więcej informacji na temat zachowanych publikacji zawiera opis PMRET), ale nie jest używana jako parametr MDAOD , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ we wszystkich wysłanych do nich publikacjach podano wartość przesłaniającą parametr MDAOD . Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez LNAORD. Wartością początkową tego pola są 4 znaki odstępu.

MDBOC (10-cyfrowa liczba całkowita ze znakiem)

Licznik wycofania.

Jest to liczba wskazująca, ile razy komunikat został poprzednio zwrócony przez wywołanie MQGET jako część jednostki pracy, a następnie wycofany. Jest ona udostępniana aplikacji jako pomoc w wykrywaniu błędów przetwarzania, które są oparte na treści komunikatu. Liczba wykluczeń wywołań MQGET, które określały dowolną z opcji GMBRW*.

Na dokładność tej liczby ma wpływ atrybut kolejki **HardenGetBackout** ; patrz sekcja “Atrybuty kolejek” na stronie 1410.

Jest to pole wyjściowe wywołania MQGET. Jest on ignorowany w przypadku wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest 0.

MDCID (24-bajtowy łańcuch bitowy)

Identyfikator korelacji.

Jest to łańcuch bajtów, który może zostać użyty przez aplikację do powiązania jednego komunikatu z innym lub do powiązania komunikatu z inną pracą wykonywaną przez aplikację. Identyfikator korelacji jest trwałą właściwością komunikatu i jest zachowywany po restarcie menedżera kolejek. Ponieważ identyfikator korelacji jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator korelacji nie jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do innego.

W przypadku wywołań MQPUT i MQPUT1 aplikacja może określić dowolną wartość. Menedżer kolejek przesyła tę wartość wraz z komunikatem i dostarcza ją do aplikacji, która wysyła żądanie pobrania komunikatu.

Jeśli aplikacja określa identyfikator PMNCID, menedżer kolejek generuje unikalny identyfikator korelacji, który jest wysyłany z komunikatem, a także zwracany do aplikacji wysyłającej w wyniku wywołania MQPUT lub MQPUT1 .

Ten wygenerowany identyfikator korelacji jest zachowywany wraz z komunikatem, jeśli zostanie zachowany, i używany jako identyfikator korelacji, gdy komunikat jest wysyłany jako publikacja do subskrybentów, którzy w polu SDCID w pliku MQSD przekazanym w wywołaniu MQSUB podali wartość CINONE.

Więcej informacji na temat zachowanych publikacji zawiera sekcja “MQPMO (opcje umieszczania komunikatów) w systemie IBM i” na stronie 1208 .

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole MDCID w sposób określony w polu MDREP oryginalnego komunikatu (ROCMTC lub ROPCI). Należy to zrobić również w przypadku aplikacji generujących komunikaty raportów.

W przypadku wywołania MQGET MDCID jest jednym z pięciu pól, za pomocą których można wybrać konkretny komunikat do pobrania z kolejki. Szczegółowe informacje na temat określania wartości dla tego pola zawiera opis pola MDMID .

Określenie wartości CINONE jako identyfikatora korelacji ma taki sam efekt, jak nieokreślenie wartości MOCORI, co oznacza, że każdy identyfikator korelacji będzie zgodny.

Jeśli opcja GMMUC jest określona w parametrze **GMO** wywołania MQGET, to pole jest ignorowane.

Po powrocie z wywołania MQGET pole MDCID jest ustawiane na identyfikator korelacji zwróconego komunikatu (jeśli istnieje).

Można stosować następujące wartości specjalne:

KINON
Nie określono identyfikatora korelacji.

Wartością długości pola jest zero binarne.

CINEWS
Komunikat jest początkiem nowej sesji.

Ta wartość jest rozpoznawana przez CICS bridge jako początek nowej sesji, czyli początek nowej sekwencji komunikatów.

W przypadku wywołania MQGET jest to pole wejściowe/wyjściowe. W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe, jeśli nie określono identyfikatora PMNCID, oraz pole wyjściowe, jeśli określono identyfikator PMNCID. Długość tego pola jest określona przez LNCID. Wartością początkową tego pola jest CINONE.

MDCSI (10-cyfrowa liczba całkowita ze znakiem)

Określa identyfikator zestawu znaków danych znakowych w komunikacie.

Uwaga: Dane znakowe w strukturze MQMD i innych strukturach danych IBM MQ , które są parametrami w wywołaniach, muszą znajdować się w zestawie znaków menedżera kolejek. Jest on

definiowany przez atrybut **CodedCharSetId** menedżera kolejek. Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty menedżera kolejek w systemie IBM i” na stronie 1443 .

Można użyć następujących wartości specjalnych:

CSQM
Identyfikator zestawu znaków menedżera kolejek.

Dane znakowe w komunikacie znajdują się w zestawie znaków menedżera kolejek.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia tę wartość w deskryptorze MQMD wysłanym z komunikatem na prawdziwy identyfikator zestawu znaków menedżera kolejek. W rezultacie wartość CSQM nigdy nie jest zwracana przez wywołanie MQGET.

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w komunikacie znajdują się w tym samym zestawie znaków, co ta struktura. Jest to zestaw znaków menedżera kolejek. (Tylko w przypadku MQMD, CSINHT ma takie samo znaczenie jak CSQM).

Menedżer kolejek zmienia tę wartość w strukturze MQMD wysłanej z komunikatem na rzeczywisty identyfikator zestawu znaków MQMD. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola MDPAT w MQMD jest ATBRKR.

CSEMBD,
Identyfikator osadzonego zestawu znaków.

Dane znakowe w komunikacie znajdują się w zestawie znaków z identyfikatorem zawartym w samych danych komunikatu. W danych komunikatu może być osadzona dowolna liczba identyfikatorów zestawów znaków, które mają zastosowanie do różnych części danych. Ta wartość musi być używana dla komunikatów PCF, które zawierają dane w mieszance zestawów znaków. Komunikaty PCF mają nazwę formatu FMPCF.

Tę wartość należy podać tylko w wywołaniach MQPUT i MQPUT1 . Jeśli jest ona określona w wywołaniu MQGET, uniemożliwia konwersję komunikatu.

W wywołaniach MQPUT i MQPUT1 menedżer kolejek zmienia wartości CSQM i CSINHT w deskryptorze MQMD wysłanym z komunikatem zgodnie z wcześniejszym opisem, ale nie zmienia deskryptora MQMD określonego w wywołaniu MQPUT lub MQPUT1 . Dla podanej wartości nie jest wykonywane żadne inne sprawdzenie.

Aplikacje, które pobierają komunikaty, powinny porównać to pole z wartością oczekiwaną przez aplikację. Jeśli wartości różnią się, aplikacja może potrzebować konwersji danych znakowych w komunikacie.

Jeśli w wywołaniu MQGET określono opcję GMCONV, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest identyfikatorem kodowanego zestawu znaków, na który w razie potrzeby powinny zostać przekształcone dane komunikatu. Jeśli konwersja powiedzie się lub nie będzie potrzebna, wartość pozostanie niezmieniona (z wyjątkiem tego, że wartość CSQM lub CSINHT zostanie przekształcona w wartość rzeczywistą). Jeśli konwersja nie powiedzie się, wartość po wywołaniu MQGET reprezentuje identyfikator kodowanego zestawu znaków nieprzekształconego komunikatu, który jest zwracany do aplikacji.

W przeciwnym razie jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest CSQM.

MDENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe danych komunikatu.

Określa kodowanie liczbowe danych liczbowych w komunikacie. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQMD. Kodowanie liczbowe definiuje reprezentację używaną dla binarnych liczb całkowitych, upakowanych liczb całkowitych i liczb zmiennopozycyjnych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Zdefiniowana jest następująca wartość specjalna:

ENNAT,
Kodowanie na komputerze rodzimym.

Kodowanie jest domyślne dla języka programowania i komputera, na którym działa aplikacja.

Uwaga: Wartość tej stałej zależy od języka programowania i środowiska. Z tego powodu aplikacje muszą być kompilowane przy użyciu plików nagłówka, makra, COPY lub INCLUDE odpowiednich dla środowiska, w którym aplikacja będzie uruchamiana.

Aplikacje, które umieszczają komunikaty, powinny zwykle określać ENNAT. Aplikacje, które pobierają komunikaty, powinny porównać to pole z wartością ENNAT. Jeśli wartości różnią się, aplikacja może potrzebować konwersji danych liczbowych w komunikacie. Za pomocą opcji GMCONV można zażądać od menedżera kolejek przekształcenia komunikatu w ramach przetwarzania wywołania MQGET.

Jeśli w wywołaniu MQGET określono opcję GMCONV, to pole jest polem wejścia/wyjścia. Wartość określona przez aplikację jest kodowaniem, na które w razie potrzeby powinny zostać przekształcone dane komunikatu. Jeśli konwersja powiedzie się lub nie będzie potrzebna, wartość pozostanie niezmieniona. Jeśli konwersja nie powiedzie się, wartość po wywołaniu MQGET reprezentuje kodowanie nieprzekształconego komunikatu, który jest zwracany do aplikacji.

W innych przypadkach jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest ENNAT.

MDEXP (10-cyfrowa liczba całkowita ze znakiem)

Czas życia komunikatu.

Jest to okres czasu wyrażony w dziesiątych częściach sekundy, ustawiony przez aplikację, która umieszcza komunikat. Komunikat zostanie zakwalifikowany do usunięcia, jeśli nie został usunięty z kolejki docelowej przed upływem tego czasu.

Wartość jest zmniejszana w celu odzwierciedlenia czasu, jaki komunikat spędza w kolejce docelowej, a także w pośrednich kolejkach transmisji, jeśli operacja put jest w kolejce zdalnej. Wartość ta może być również zmniejszana przez agenty kanału komunikatów w celu odzwierciedlenia czasów transmisji, jeśli są one istotne. Podobnie aplikacja przekazująca ten komunikat do innej kolejki może w razie potrzeby zmniejszyć tę wartość, jeśli zachowała komunikat przez dłuższy czas. Jednak czas ważności jest traktowany jako przybliżony, a wartość nie musi być zmniejszana w celu odzwierciedlenia niewielkich przedziałów czasu.

Gdy komunikat jest pobierany przez aplikację przy użyciu wywołania MQGET, pole MDEXP reprezentuje czas pierwotnej utraty ważności, który nadal pozostaje.

Po upływie czasu utraty ważności komunikatu zostaje on zakwalifikowany do usunięcia przez menedżer kolejek. W bieżących implementacjach komunikat jest odrzucany, gdy wystąpi wywołanie MQGET przeglądania lub nieprzeglądania, które zwróciłoby komunikat, gdyby nie utracił ważności. Na przykład nieprzeglądanie wywołania MQGET z polem GMMO w obiekcie MQGMO ustawionym na wartość MONONE podczas odczytu z kolejki uporządkowanej FIFO spowoduje usunięcie wszystkich komunikatów, które utraciły ważność, aż do pierwszego komunikatu, który nie utracił ważności. W przypadku kolejki uporządkowanej według priorytetu to samo wywołanie spowoduje usunięcie komunikatów o wyższym priorytecie, które utraciły ważność, oraz komunikatów o takim samym priorytecie, które dotarły do kolejki przed pierwszym komunikatem, który nie utracił ważności.

Komunikat, który utracił ważność, nigdy nie jest zwracany do aplikacji (ani przez przeglądanie, ani przez wywołanie MQGET bez przeglądania), więc wartość w polu MDEXP deskryptora komunikatu po pomyślnym wywołaniu MQGET jest większa niż zero lub wartość specjalna EIULIM.

Jeśli komunikat jest umieszczany w kolejce zdalnej, może utracić ważność (i zostać odrzucony), gdy znajduje się w pośredniej kolejce transmisji, zanim komunikat dotrze do kolejki docelowej.

Raport jest generowany, gdy komunikat, który utracił ważność, zostanie odrzucony, jeśli w komunikacie określono jedną z opcji raportu ROEXP*. Jeśli nie zostanie podana żadna z tych opcji,

nie zostanie wygenerowany taki raport. Komunikat nie będzie już istotny po upływie tego okresu (być może dlatego, że został zastąpiony przez późniejszy komunikat).

Każdy inny program, który usuwa komunikaty na podstawie czasu utraty ważności, musi również wysłać odpowiedni komunikat raportu, jeśli został zażądany.

Uwaga:

1. Jeśli komunikat jest umieszczany z wartością MDEXP równą zero, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny RC2013; w tym przypadku nie jest generowany żaden komunikat raportu.
2. Ponieważ komunikat z czasem utraty ważności, który upłynął, może nie zostać usunięty aż do późniejszego momentu, w kolejce mogą znajdować się komunikaty, które przekroczyli swój czas utraty ważności i dlatego nie kwalifikują się do pobrania. Komunikaty te są jednak wliczane do liczby komunikatów w kolejce dla wszystkich celów, łącznie z wyzwalaniem głębokości.
3. Raport o utracie ważności jest generowany, jeśli zostanie zgłoszony, gdy komunikat zostanie rzeczywiście odrzucony, a nie wtedy, gdy zostanie zakwalifikowany do usunięcia.
4. Usunięcie komunikatu, który utracił ważność, i wygenerowanie raportu o utracie ważności, jeśli jest to żądane, nigdy nie są częścią jednostki pracy aplikacji, nawet jeśli komunikat został zaplanowany do usunięcia w wyniku wywołania MQGET działającego w ramach jednostki pracy.
5. Jeśli komunikat, który prawie utracił ważność, jest pobierany przez wywołanie MQGET w obrębie jednostki pracy, a następnie jednostka pracy jest wycofywana, komunikat może zostać zakwalifikowany do usunięcia przed ponownym pobraniem.
6. Jeśli prawie wygasły komunikat jest zablokowany przez wywołanie MQGET z GMLK, komunikat może zostać zakwalifikowany do usunięcia przed pobraniem go przez wywołanie MQGET z GMMUC; w takim przypadku w następnym wywołaniu MQGET zwracany jest kod przyczyny RC2034 .
7. Po pobraniu komunikatu żądania z czasem utraty ważności większym niż zero aplikacja może wykonać jedno z następujących działań podczas wysyłania komunikatu odpowiedzi:
   - Skopiuj pozostały czas ważności z komunikatu żądania do komunikatu odpowiedzi.
   - Ustaw czas utraty ważności w komunikacie odpowiedzi na wartość jawną większą niż zero.
   - Ustaw czas utraty ważności w komunikacie odpowiedzi na EIULIM.

   Działanie, które należy wykonać, zależy od projektu pakietu aplikacji. Jednak domyślnym działaniem dla umieszczania komunikatów w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) powinno być zachowanie pozostałego czasu utraty ważności komunikatu i dalsze zmniejszanie tego czasu.
8. Komunikaty wyzwalacza są zawsze generowane z EIULIM.
9. Komunikat (zwykle w kolejce transmisji) o nazwie MDFMT FMXQH ma drugi deskryptor komunikatu w MQXQH. Dlatego jest z nim powiązane dwa pola MDEXP . W tym przypadku należy zwrócić uwagę na następujące dodatkowe kwestie:
   - Gdy aplikacja umieszcza komunikat w kolejce zdalnej, menedżer kolejek umieszcza komunikat początkowo w lokalnej kolejce transmisji i poprzedza dane komunikatu aplikacji strukturą MQXQH. Menedżer kolejek ustawia wartości dwóch pól MDEXP na takie same, jak te określone przez aplikację.

     Jeśli aplikacja umieszcza komunikat bezpośrednio w lokalnej kolejce transmisji, dane komunikatu muszą już zaczynać się od struktury MQXQH, a nazwą formatu musi być FMXQH (ale menedżer kolejek nie wymusza tego). W takim przypadku aplikacja nie musi ustawiać jednakowe wartości tych dwóch pól MDEXP . (Menedżer kolejek nie sprawdza, czy pole MDEXP w MQXQH zawiera poprawną wartość, a nawet czy dane komunikatu są wystarczająco długie, aby je dołączyć).
   - Gdy komunikat o nazwie MDFMT FMXQH jest pobierany z kolejki (bez względu na to, czy jest to kolejka normalna, czy kolejka transmisji), menedżer kolejek zmniejsza wartość obu tych pól MDEXP o czas oczekiwania w kolejce. Jeśli dane komunikatu nie są wystarczająco długie, aby uwzględnić pole MDEXP w MQXQH, nie jest zgłaszany żaden błąd.

- Menedżer kolejek używa pola MDEXP w osobnym deskryptorze komunikatu (czyli nie w deskryptorze komunikatu osadzonym w strukturze MQXQH) do sprawdzenia, czy komunikat kwalifikuje się do usunięcia.
- Jeśli wartości początkowe dwóch pól MDEXP były różne, czas MDEXP w oddzielnym deskryptorze komunikatu, gdy komunikat jest pobierany, może być większy od zera (dlatego komunikat nie jest zakwalifikowany do usunięcia), podczas gdy upłynął czas zgodny z wartością pola MDEXP w MQXQH. W tym przypadku pole MDEXP w MQXQH jest ustawione na wartość zero.

Rozpoznawana jest następująca wartość specjalna:

EIULIM
Nieograniczony czas życia.

Komunikat ma nieograniczony czas ważności.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest EIULIM.

MDFB (10-cyfrowa liczba całkowita ze znakiem)

Informacja zwrotna lub kod przyczyny.

Jest ona używana z komunikatem typu MTRPRT w celu wskazania rodzaju raportu i ma znaczenie tylko w przypadku tego typu komunikatu. Pole może zawierać jedną z wartości FB* lub jedną z wartości RC*. Kody sprzężenia zwrotnego są pogrupowane w następujący sposób:

FBNONE
Nie podano informacji zwrotnej.

FBSFST
Najniższa wartość informacji zwrotnej generowanej przez system.

FBSLST
Najwyższa wartość dla informacji zwrotnych wygenerowanych przez system.

Zakres wygenerowanych przez system kodów sprzężenia zwrotnego od FBSFST do FBSLST obejmuje ogólne kody sprzężenia zwrotnego wymienione w dalszej części tej sekcji (FB*), a także kody przyczyny (RC*), które mogą wystąpić, gdy komunikat nie może zostać umieszczony w kolejce docelowej.

FBAFST
Najniższa wartość dla informacji zwrotnych wygenerowanych przez aplikację.

FBALST
Najwyższa wartość dla informacji zwrotnych wygenerowanych przez aplikację.

Aplikacje generujące komunikaty raportu nie powinny używać kodów sprzężenia zwrotnego z zakresu systemu (innego niż FBQUIT), chyba że chcą symulować komunikaty raportu wygenerowane przez menedżera kolejek lub agenta kanału komunikatów.

W wywołaniach MQPUT lub MQPUT1 podana wartość musi być równa FBNONE lub należeć do zakresu systemu lub zakresu aplikacji. Ta opcja jest sprawdzana niezależnie od wartości parametru MDMT.

Ogólne kody sprzężenia zwrotnego:

FBCOA
Potwierdzenie przybycia do kolejki docelowej (patrz ROCOA).

FBCOD
Potwierdzenie dostarczenia do aplikacji przyjmującej (zob. ROCOD).

FBEXP
Komunikat utracił ważność.

Komunikat został odrzucony, ponieważ nie został usunięty z kolejki docelowej przed upływem czasu ważności.

FBPAN
Powiadomienie o działaniu pozytywnym (zob. ROPAN).

FBNAN
Powiadomienie o negatywnym działaniu (patrz RONAN).

FBQUIT
Aplikacja powinna zostać zakończona.

Może być ona używana przez program do planowania obciążenia do sterowania liczbą uruchomionych instancji aplikacji. Wysłanie komunikatu MTRPRT z tym kodem informacji zwrotnej do instancji aplikacji wskazuje tej instancji, że powinna ona zatrzymać przetwarzanie. Jednak przestrzeganie tej konwencji jest sprawą aplikacji i nie jest wymuszane przez menedżer kolejek.

IMS-bridge feedback codes: Gdy most IMS otrzyma niezerowy kod rozpoznania IMS-OTMA, most IMS przekształca kod rozpoznania z szesnastkowego na dziesiętny, dodaje wartość FBIERR (300) i umieszcza wynik w polu MDFB komunikatu odpowiedzi. Powoduje to, że kod sprzężenia zwrotnego ma wartość z zakresu od FBIFST (301) do FBILST (399), gdy wystąpił błąd IMS-OTMA.

Most IMS może wygenerować następujące kody sprzężenia zwrotnego:

FBDLZ,
Zerowa długość danych.

Długość segmentu w danych aplikacji komunikatu wynosiła zero.

FBDLN,
Ujemna długość danych.

Długość segmentu była ujemna w danych aplikacji komunikatu.

FBDLTB
Zbyt duża długość danych.

Długość segmentu była zbyt duża w danych aplikacji komunikatu.

FBBUFO,
Przepełnienie buforu.

Wartość jednego z pól długości spowodowałaby przepełnienie buforu komunikatów przez dane.

FBLOB1
Błędna długość o jeden.

Wartość jednego z pól długości była o jeden bajt za krótka.

FBIIH
Struktura MQIIH jest niepoprawna lub nie istnieje.

Pole MDFMT w strukturze MQMD określa wartość FMIMS, ale komunikat nie rozpoczyna się od poprawnej struktury MQIIH.

FBNAFI
ID użytkownika nie ma uprawnień do użycia w programie IMS.

Sprawdzenie poprawności przez most IMS nie powiodło się dla identyfikatora użytkownika zawartego w deskryptorze komunikatu MQMD lub hasła zawartego w polu IIAUT w strukturze MQIIH. W rezultacie komunikat nie został przekazany do programu IMS.

FBIERR
Program IMSzwrócił nieoczekiwany błąd.

IMSzwrócił nieoczekiwany błąd. Więcej informacji na temat błędu można znaleźć w dzienniku błędów systemu IBM MQ , w którym znajduje się most IMS .

FBIFST
Najniższa wartość dla wygenerowanej przez IMSopinii.

Wygenerowane przez IMSkody sprzężenia zwrotnego zajmują zakres od FBIFST (300) do FBILST (399). Sam kod diagnostyczny IMS-OTMA to MDFB minus FBIERR.

FBILST
Najwyższa wartość dla opinii generowanej przez IMS.

CICS-bridge feedback codes: CICS bridgemoże wygenerować następujące kody sprzężenia zwrotnego:

FBCAAB
Aplikacja została zakończona awaryjnie.

Aplikacja określona w komunikacie została zakończona awaryjnie. Ten kod sprzężenia zwrotnego występuje tylko w polu DLREA struktury MQDLH.

FBCANS
Nie można uruchomić aplikacji.

Operacja EXEC CICS LINK dla aplikacji określonej w komunikacie nie powiodła się. Ten kod sprzężenia zwrotnego występuje tylko w polu DLREA struktury MQDLH.

FBCBRF,
Program CICS bridge został zakończony nieprawidłowo bez zakończenia normalnego przetwarzania błędów.

FBCCSE
Niepoprawny identyfikator zestawu znaków.

FBCIHE
Brak lub niepoprawna struktura nagłówka informacji CICS .

FBCCAE,
Niepoprawna długość obszaru komunikacyjnego CICS .

FBCCIE
Niepoprawny identyfikator korelacji.

BBCDLQ
Kolejka niedostarczonych komunikatów jest niedostępna.

Zadanie CICS bridge nie mogło skopiować odpowiedzi na to żądanie do kolejki niedostarczonych komunikatów. Żądanie zostało wycofane.

FBCENE
Niepoprawne kodowanie.

FBCINNA
Program CICS bridge napotkał nieoczekiwany błąd.

Ten kod sprzężenia zwrotnego występuje tylko w polu DLREA struktury MQDLH.

FBCNTA
Identyfikator użytkownika nie jest autoryzowany lub hasło jest niepoprawne.

Ten kod sprzężenia zwrotnego występuje tylko w polu DLREA struktury MQDLH.

FBCUOB
Jednostka pracy wycofała się.

Jednostka pracy została wycofana z jednego z następujących powodów:

- Wykryto niepowodzenie podczas przetwarzania innego żądania w tej samej jednostce pracy.
- W trakcie trwania jednostki pracy wystąpiło nieprawidłowe zakończenie CICS .

FBCUWE
Niepoprawne pole sterujące jednostki pracy CIUOW .

Kody przyczyny produktuMQ: w przypadku komunikatów raportu o wyjątkach MDFB zawiera kod przyczyny produktu MQ . Możliwe są następujące kody przyczyny:

RC2051
(2051, X'803 ') Wywołania umieszczania zablokowane dla kolejki.

RC2053
(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

RC2035
(2035, X'7F3') Brak uprawnień dostępu.

RC2056
(2056, X'808 ') Brak miejsca na dysku dla kolejki.

RC2048
(2048, X'800 ') Kolejka nie obsługuje komunikatów trwałych.

RC2031
(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

RC2030
(2030, X'7EE') Długość komunikatu przekracza maksimum dla kolejki.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest FBNONE.

MDFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych komunikatu.

Jest to nazwa, której nadawca komunikatu może użyć do wskazania odbiorcy rodzaju danych w komunikacie. Dla nazwy można określić dowolne znaki należące do zestawu znaków menedżera kolejek, ale zaleca się, aby nazwa ta była ograniczona do następujących elementów:

- Wielkie litery od A do Z
- Cyfry od 0 do 9

Jeśli używane są inne znaki, translacja nazwy między zestawami znaków menedżerów kolejek wysyłających i odbierających może być niemożliwa.

Nazwa powinna być dopełniona spacjami do długości pola lub znakiem o kodzie zero używanym do zakończenia nazwy przed końcem pola; znak o kodzie zero i wszystkie kolejne znaki są traktowane jako odstępy. Nie należy podawać nazwy z odstępami wiodącymi lub osadzonymi. W przypadku wywołania MQGET menedżer kolejek zwraca nazwę dopełnianą spacjami do długości pola.

Menedżer kolejek nie sprawdza, czy nazwa jest zgodna z zaleceniami opisanymi wcześniej.

Nazwy rozpoczynające się od łańcuchaMQ(wielkie, małe i mieszane) mają znaczenie zdefiniowane przez menedżer kolejek. W przypadku własnych formatów nie należy używać nazw rozpoczynających się od tych liter. Wbudowane formaty menedżera kolejek są następujące:

BRAK FMNONE
Brak nazwy formatu.

Rodzaj danych jest niezdefiniowany. Oznacza to, że danych nie można przekształcić, gdy komunikat jest wczytywany z kolejki przy użyciu opcji GMCONV.

Jeśli w wywołaniu MQGET określono wartość GMCONV, a zestaw znaków lub kodowanie danych w komunikacie różnią się od kodowania określonego w parametrze **MSGDSC** , komunikat jest zwracany z następującymi kodami zakończenia i przyczyny (przy założeniu braku innych błędów):

- Kod zakończenia CCWARN i kod przyczyny RC2110 , jeśli dane FMNONE znajdują się na początku komunikatu.
- Kod zakończenia CCOK i kod przyczyny RCNONE, jeśli dane FMNONE znajdują się na końcu komunikatu (czyli są poprzedzone co najmniej jedną strukturą nagłówka produktu MQ). W tym przypadku struktury nagłówków MQ są przekształcane w żądany zestaw znaków i kodowanie.

FMADMN
Komunikat żądania/odpowiedzi serwera komend.

Komunikat jest żądaniem serwera komend lub komunikatem odpowiedzi w formacie komend programowalnych (PCF). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV. Więcej informacji na temat korzystania z komunikatów

w formacie komend programowalnych zawiera sekcja Korzystanie z formatów komend programowalnych.

FMCICS
CICS .

Dane komunikatu rozpoczynają się od nagłówka informacji CICS MQCIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole CIFMT w strukturze MQCIH.

FMCMD1
Wpisz 1 komunikat odpowiedzi na komendę.

Komunikat jest komunikatem odpowiedzi serwera komend MQSC zawierającym liczbę obiektów, kod zakończenia i kod przyczyny. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMCMD2
Wpisz 2 komunikat odpowiedzi na komendę.

Komunikat jest komunikatem odpowiedzi serwera komend MQSC zawierającym informacje o żądanych obiektach. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMDLH
Nagłówek niedostarczonego komunikatu.

Dane komunikatu rozpoczynają się od nagłówka MQDLH niedostarczonego komunikatu. Dane z oryginalnego komunikatu są bezpośrednio zgodne ze strukturą MQDLH. Nazwa formatu oryginalnych danych komunikatu jest podana w polu DLFMT w strukturze MQDLH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQDLH (nagłówek niedostarczonego komunikatu) w systemie IBM i” na stronie 1094 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

Raporty COA i COD nie są generowane dla komunikatów, które mają parametr MDFMT o wartości FMDLH.

FMDH
Nagłówek listy dystrybucyjnej.

Dane komunikatu rozpoczynają się od nagłówka listy dystrybucyjnej MQDH. Obejmuje to tablice rekordów MQOR i MQPMR. Po nagłówku listy dystrybucyjnej mogą następować dodatkowe dane. Format dodatkowych danych (jeśli istnieją) jest podany w polu DHFMT w strukturze MQDH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQDH (nagłówek dystrybucji) w systemie IBM i” na stronie 1089 . Komunikaty w formacie FMDH mogą być przekształcane, jeśli w wywołaniu MQGET podano opcję GMCONV.

FMEVNT,
Komunikat zdarzenia.

Komunikat jest komunikatem zdarzenia MQ , który zgłasza wystąpienie zdarzenia. Komunikaty zdarzeń mają taką samą strukturę jak komendy programowalne. Więcej informacji na temat tej struktury zawiera sekcja Struktury komend i odpowiedzi. Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Komunikaty zdarzeń w wersji Version-1 mogą być przekształcane, jeśli w wywołaniu MQGET podano opcję GMCONV.

FMIMS
IMS .

Dane komunikatu rozpoczynają się od nagłówka informacji IMS MQIIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest podawana przez pole *IIFMT* w strukturze MQIIH. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMIMVS,
IMS łańcuch zmiennej.

Komunikat jest łańcuchem zmiennej IMS , który jest łańcuchem w postaci `llzzccc`, gdzie:

ll
to pole o długości 2 bajtów określające całkowitą długość elementu łańcucha zmiennej IMS . Ta długość jest równa długości `ll` (2 bajty) plus długości `zz` (2 bajty) plus długości samego łańcucha znaków. `ll` jest 2-bajtową binarną liczbą całkowitą w kodowaniu określonym przez pole `MDENC` .

zz
to pole dwubajtowe zawierające flagi istotne dla IMS. `zz` jest łańcuchem bajtowym składającym się z dwóch 1-bajtowych pól łańcucha i przesyłanym bez zmiany nadawcy na odbiorcę (czyli `zz` nie podlega żadnej konwersji).

ccc
jest łańcuchem znaków o zmiennej długości, zawierającym znaki `ll-4` . `ccc` znajduje się w zestawie znaków określonym w polu `MDCSI` .

Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMMDE,
Rozszerzenie deskryptora komunikatu.

Dane komunikatu rozpoczynają się od rozszerzenia deskryptora komunikatu MQMDE i opcjonalnie następują po nich inne dane (zazwyczaj dane komunikatu aplikacji). Nazwa formatu, zestaw znaków i kodowanie danych zgodne z MQMDE są nadawane przez pola `MEFMT`, `MECSI`i `MEENC` w MQMDE. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQMDE (Rozszerzenie deskryptora komunikatu) w systemie IBM i” na stronie 1187 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMPCF,
Komunikat zdefiniowany przez użytkownika w formacie komend programowalnych (PCF).

Komunikat jest komunikatem zdefiniowanym przez użytkownika, który jest zgodny ze strukturą komunikatu PCF (programmable command format). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV. Więcej informacji na temat korzystania z komunikatów w formacie komend programowalnych zawiera sekcja Korzystanie z formatów komend programowalnych .

FMRMH
Nagłówek komunikatu odniesienia.

Dane komunikatu rozpoczynają się od nagłówka komunikatu odniesienia MQRMH, po którym opcjonalnie następują inne dane. Nazwa formatu, zestaw znaków i kodowanie danych są nadawane przez pola `RMFMT`, `RMCSI`i `RMENC` w MQRMH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQRMH (nagłówek komunikatu odniesienia) w systemie IBM i” na stronie 1235 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMRFH
Reguły i nagłówek formatowania.

Dane komunikatu rozpoczynają się od nagłówka MQRFH reguł i formatowania, po którym opcjonalnie następują inne dane. Nazwa formatu, zestaw znaków i kodowanie danych (jeśli istnieją) są nadawane przez pola `RFFMT`, `RFCSI`i `RFENC` w MQRFH. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMRFH2
Reguły i formatowanie nagłówka w wersji 2.

Dane komunikatu rozpoczynają się od reguł version-2 i nagłówka formatowania MQRFH2, po którym opcjonalnie następują inne dane. Nazwa formatu, zestaw znaków i kodowanie danych

opcjonalnych (jeśli istnieją) są nadawane przez pola RF2FMT, RF2CSIi RF2ENC w nagłówku MQRFH2. Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMSTR,
Komunikat składający się w całości ze znaków.

Dane komunikatu aplikacji mogą być łańcuchem SBCS (zestaw znaków jednobajtowych) lub łańcuchem DBCS (zestaw znaków dwubajtowych). Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FMTM,
Komunikat wyzwalacza.

Komunikat jest komunikatem wyzwalacza opisanym przez strukturę MQTM. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQTM-komunikat wyzwalacza” na stronie 1273 . Komunikaty w tym formacie mogą być przekształcane, jeśli w wywołaniu MQGET określono opcję GMCONV.

FWIH
Nagłówek informacji o pracy.

Dane komunikatu rozpoczynają się od nagłówka informacji o pracy MQWIH, po którym następują dane aplikacji. Nazwa formatu danych aplikacji jest nadawana przez pole WIFMT w strukturze MQWIH.

FMXQH
Nagłówek kolejki transmisji.

Dane komunikatu rozpoczynają się od nagłówka kolejki transmisji MQXQH. Dane z oryginalnego komunikatu są bezpośrednio zgodne ze strukturą MQXQH. Nazwa formatu oryginalnych danych komunikatu jest nadawana przez pole MDFMT w strukturze MQMD, które jest częścią nagłówka kolejki transmisji MQXQH. Szczegółowe informacje na temat tej struktury zawiera sekcja “MQXQH (nagłówek kolejki transmisji) w systemie IBM i” na stronie 1284 .

Raporty COA i COD nie są generowane dla komunikatów, które mają MDFMT FMXQH.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest określona przez LNFMT. Wartością początkową tego pola jest FMNONE.

MDGID (24-bajtowy łańcuch bitowy)

Identyfikator grupy.

Jest to łańcuch bajtów używany do identyfikowania konkretnej grupy komunikatów lub komunikatu logicznego, do którego należy komunikat fizyczny. Parametr MDGID jest również używany, jeśli dla komunikatu dozwolona jest segmentacja. We wszystkich tych przypadkach MDGID ma wartość inną niż null i w polu MDMFL jest ustawiona co najmniej jedna z następujących flag:

- MMIG
- MFLMIG@ item: indo
- MSEG
- MFLSEG
- MSEGA

Jeśli żadna z tych opcji nie jest ustawiona, MDGID ma specjalną wartość null GINONE.

To pole nie musi być ustawiane przez aplikację w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono PMLOGO.
- W wywołaniu MQGET nie określono parametru MOGRPI.

Należy rozważyć użycie tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołanie ma wartość MQPUT1, aplikacja musi upewnić się, że parametr MDGID ma odpowiednią wartość.

Grupy komunikatów i segmenty mogą być poprawnie przetwarzane tylko wtedy, gdy identyfikator grupy jest unikalny. Z tego powodu aplikacje nie powinny generować własnych identyfikatorów grup; zamiast tego aplikacje powinny wykonywać jedną z następujących czynności:

- Jeśli określono PMLOGO, menedżer kolejek automatycznie generuje unikalny identyfikator grupy dla pierwszego komunikatu w grupie lub segmencie komunikatu logicznego i używa tego identyfikatora grupy dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego, więc aplikacja nie musi podejmować żadnych działań specjalnych. Należy rozważyć zastosowanie tej procedury.
- Jeśli parametr PMLOGO nie jest określony, aplikacja powinna zażądać od menedżera kolejek wygenerowania identyfikatora grupy, ustawiając parametr MDGID na wartość GINONE w pierwszym wywołaniu MQPUT lub MQPUT1 dla komunikatu w grupie lub segmencie komunikatu logicznego. Identyfikator grupy zwracany przez menedżera kolejek w danych wyjściowych tego wywołania powinien być następnie używany dla pozostałych komunikatów w grupie lub segmentach komunikatu logicznego. Jeśli grupa komunikatów zawiera komunikaty posegmentowane, ten sam identyfikator grupy musi być używany dla wszystkich segmentów i komunikatów w grupie.

  Jeśli parametr PMLOGO nie jest określony, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności (na przykład w kolejności odwrotnej), ale identyfikator grupy musi być przydzielony przez pierwsze wywołanie MQPUT lub MQPUT1 , które jest wysyłane dla dowolnego z tych komunikatów.

Na wejściu wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości określonej w sekcji PMOPT. W danych wyjściowych wywołań MQPUT i MQPUT1 menedżer kolejek ustawia w tym polu wartość, która została wysłana z komunikatem, jeśli otwarty obiekt jest pojedynczą kolejką, a nie listą dystrybucyjną, ale pozostawia to pole bez zmian, jeśli otwarty obiekt jest listą dystrybucyjną. W tym drugim przypadku, jeśli aplikacja musi znać wygenerowane identyfikatory grup, musi udostępnić rekordy MQPMR zawierające pole PRGID .

Na wejściu wywołania MQGET menedżer kolejek używa wartości określonej w Tabeli 1. W danych wyjściowych wywołania MQGET menedżer kolejek ustawia w tym polu wartość pobranego komunikatu.

Zdefiniowana jest następująca wartość specjalna:

GINON
Nie określono identyfikatora grupy.

Wartością długości pola jest zero binarne. Jest to wartość używana dla komunikatów, które nie są w grupach, nie są segmentami komunikatów logicznych i dla których segmentacja nie jest dozwolona.

Długość tego pola jest określona przez LNGID. Wartością początkową tego pola jest GINONE. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDMFL (10-cyfrowa liczba całkowita ze znakiem)

Flagi komunikatów.

Są to flagi, które określają atrybuty komunikatu lub sterują jego przetwarzaniem. Flagi są podzielone na następujące kategorie:

- Flaga segmentacji
- Flagi statusu

Są one z kolei opisane.

Opcje segmentacji: Jeśli komunikat jest zbyt duży dla kolejki, próba umieszczenia komunikatu w kolejce zwykle kończy się niepowodzeniem. Segmentacja to technika, w której menedżer kolejek lub aplikacja dzieli komunikat na mniejsze części nazywane segmentami i umieszcza każdy segment w kolejce jako oddzielny komunikat fizyczny. Aplikacja, która pobiera komunikat, może pobrać segmenty pojedynczo lub zażądać od menedżera kolejek ponownego złożenia segmentów w pojedynczy komunikat, który jest zwracany przez wywołanie MQGET. Ten ostatni jest osiągany przez określenie opcji GMCMPM w wywołaniu MQGET i podanie buforu, który jest wystarczająco duży, aby pomieścić cały komunikat. (Szczegółowe informacje na temat opcji GMCMPM zawiera sekcja “MQGMO

(opcje pobierania komunikatów) w systemie IBM i" na stronie 1106). Segmentacja komunikatu może wystąpić w nadawczym menedżerze kolejek, w pośrednim menedżerze kolejek lub w docelowym menedżerze kolejek.

Aby sterować segmentacją komunikatu, można określić jedną z następujących opcji:

MSEGI
Segmentacja zablokowana.

Ta opcja zapobiega rozbiciu komunikatu na segmenty przez menedżer kolejek. Jeśli ta opcja jest określona dla komunikatu, który jest już segmentem, zapobiega rozbiciu segmentu na mniejsze segmenty.

Wartością tej flagi jest zero binarne. Jest to opcja domyślna.

MSEGA
Segmentacja jest dozwolona.

Ta opcja umożliwia podzieleniem komunikatu na segmenty przez menedżer kolejek. Jeśli ta opcja jest określona dla komunikatu, który jest już segmentem, umożliwia podział segmentu na mniejsze segmenty. MFSEGA można ustawić bez ustawiania MFSEG lub MFLSEG.

Gdy menedżer kolejek segmentuje komunikat, menedżer kolejek włącza flagę MFSEG w kopii deskryptora MQMD wysyłanego z każdym segmentem, ale nie zmienia ustawień tych flag w deskryptorze MQMD udostępnianym przez aplikację w wywołaniu MQPUT lub MQPUT1 . Dla ostatniego segmentu w komunikacie logicznym menedżer kolejek włącza również flagę MFLSEG w deskryptorze MQMD wysyłanym z tym segmentem.

Uwaga: Należy zachować ostrożność, gdy komunikaty są umieszczane w MFSEGA, ale bez PMLOGO. Jeśli komunikat jest następujący:

- Nie jest segmentem, oraz
- Nie należy do grupy, oraz
- Nie jest przekazywane,

Aplikacja musi pamiętać o zresetowaniu pola `MDGID` do wartości GINONE przed każdym wywołaniem MQPUT lub MQPUT1 , aby menedżer kolejek wygenerował unikalny identyfikator grupy dla każdego komunikatu. Jeśli nie zostanie to wykonane, niepowiązane komunikaty mogą przypadkowo mieć ten sam identyfikator grupy, co może prowadzić do niepoprawnego przetwarzania. Więcej informacji o tym, kiedy należy zresetować pole `MDGID` , zawierają opisy pola `MDGID` i opcji PMLOGO.

W razie potrzeby menedżer kolejek dzieli komunikaty na segmenty w celu zapewnienia, że segmenty (oraz wszystkie dane nagłówka, które mogą być wymagane) mieszczą się w kolejce. Istnieje jednak niższy limit wielkości segmentu wygenerowanego przez menedżer kolejek i tylko ostatni segment utworzony na podstawie komunikatu może być mniejszy niż ten limit. (Dolna granica wielkości segmentu generowanego przez aplikację wynosi jeden bajt). Segmenty wygenerowane przez menedżera kolejek mogą mieć nierówną długość. Menedżer kolejek przetwarza komunikat w następujący sposób:

- Formaty zdefiniowane przez użytkownika są podzielone na granice, które są wielokrotnością 16 bajtów. Oznacza to, że menedżer kolejek nie wygeneruje segmentów mniejszych niż 16 bajtów (innych niż ostatni segment).
- Wbudowane formaty inne niż FMSTR są dzielone w punktach odpowiednich do rodzaju obecnych danych. Jednak menedżer kolejek nigdy nie dzieli komunikatu w środku struktury nagłówka MQ . Oznacza to, że segment zawierający pojedynczą strukturę nagłówka produktu MQ nie może zostać dalej podzielony przez menedżer kolejek, w związku z czym minimalna możliwa wielkość segmentu dla tego komunikatu jest większa niż 16 bajtów.

  Drugi lub późniejszy segment wygenerowany przez menedżer kolejek rozpoczyna się od jednego z następujących członów:

  – Struktura nagłówka MQ

  - Początek danych komunikatu aplikacji
  - Częściowe przejście przez dane komunikatu aplikacji
- FMSTR jest dzielone bez względu na rodzaj obecnych danych (SBCS, DBCS lub mieszane SBCS/DBCS). Jeśli łańcuch jest typu DBCS lub mieszanego SBCS/DBCS, może to spowodować, że segmenty nie będą mogły być konwertowane z jednego zestawu znaków na inny. Menedżer kolejek nigdy nie dzieli komunikatów FMSTR na segmenty mniejsze niż 16 bajtów (inne niż ostatni segment).
- Pola MDFMT, MDCSIi MDENC w strukturze MQMD każdego segmentu są ustawiane przez menedżer kolejek w celu poprawnego opisania danych znajdujących się na początku segmentu. Nazwa formatu będzie albo nazwą formatu wbudowanego, albo nazwą formatu zdefiniowanego przez użytkownika.
- Pole MDREP w strukturze MQMD segmentów z wartością MD0FF większą niż zero jest modyfikowane w następujący sposób:
  - Dla każdego typu raportu, jeśli opcja raportu ma wartość RO* D, ale segment nie może zawierać żadnego z pierwszych 100 bajtów danych użytkownika (to znaczy danych następujących po strukturach nagłówków MQ , które mogą być obecne), opcja raportu jest zmieniana na RO*.

Menedżer kolejek postępuje zgodnie z wcześniejszymi regułami, ale w przeciwnym razie dzieli komunikaty nieprzewidywalnie. Nie należy zakładać, gdzie komunikat jest dzielony.

W przypadku trwałych komunikatów menedżer kolejek może wykonać segmentację tylko w obrębie jednostki pracy:

- Jeśli wywołanie MQPUT lub MQPUT1 działa w obrębie jednostki pracy zdefiniowanej przez użytkownika, używana jest ta jednostka pracy. Jeśli wywołanie nie powiedzie się w trakcie procesu segmentacji, menedżer kolejek usuwa wszystkie segmenty, które zostały umieszczone w kolejce w wyniku wywołania zakończonego niepowodzeniem. Jednak niepowodzenie nie uniemożliwia pomyślnego zatwierdzenia jednostki pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika i nie istnieje żadna jednostka pracy zdefiniowana przez użytkownika, menedżer kolejek tworzy jednostkę pracy tylko na czas trwania wywołania. Jeśli wywołanie powiedzie się, menedżer kolejek automatycznie zatwierdza jednostkę pracy (aplikacja nie musi tego robić). Jeśli wywołanie nie powiedzie się, menedżer kolejek wycofa jednostkę pracy.
- Jeśli wywołanie działa poza jednostką pracy zdefiniowaną przez użytkownika, ale jednostka pracy zdefiniowana przez użytkownika istnieje, menedżer kolejek nie może wykonać segmentacji. Jeśli komunikat nie wymaga segmentacji, wywołanie nadal może zakończyć się powodzeniem. Jeśli jednak komunikat wymaga segmentacji, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2255.

W przypadku nietrwałych komunikatów menedżer kolejek nie wymaga, aby jednostka pracy była dostępna w celu wykonania segmentacji.

Należy zwrócić szczególną uwagę na konwersję danych komunikatów, które mogą być segmentowane:

- Jeśli konwersja danych jest wykonywana tylko przez aplikację odbierającą w wywołaniu MQGET, a aplikacja określa opcję GMCMPM, wyjście konwersji danych zostanie przekazane do kompletnego komunikatu dla wyjścia do przekształcenia, a fakt, że komunikat został segmentowany, nie będzie widoczny dla wyjścia.
- Jeśli aplikacja odbierająca pobiera po jednym segmencie naraz, wyjście konwersji danych zostanie wywołane w celu konwersji po jednym segmencie naraz. W związku z tym wyjście musi być w stanie przekształcić dane w segmencie niezależnie od danych w innych segmentach.

  Jeśli rodzaj danych w komunikacie jest taki, że dowolna segmentacja danych w granicach 16-bajtowych może spowodować, że segmenty nie będą mogły być przekształcane przez wyjście, lub jeśli formatem jest FMSTR, a zestaw znaków jest DBCS lub mieszany SBCS/DBCS, aplikacja wysyłająca powinna sama utworzyć i umieścić segmenty, określając parametr MFSEGI w celu

pominięcia dalszej segmentacji. W ten sposób aplikacja wysyłająca może zapewnić, że każdy segment zawiera informacje wystarczające do pomyślnego przekształcenia segmentu przez wyjście konwersji danych.

- Jeśli konwersja nadawcy jest określona dla agenta kanału komunikatów wysyłających (MCA), agent MCA przekształca tylko te komunikaty, które nie są segmentami komunikatów logicznych. Agent MCA nigdy nie próbuje przekształcić komunikatów, które są segmentami.

Ta opcja jest flagą wejściową wywołań MQPUT i MQPUT1 oraz flagą wyjściową wywołania MQGET. W tym drugim wywołaniu menedżer kolejek również echo wartości flagi w polu GMSEG w MQGMO.

Wartością początkową tej flagi jest MFSEGI.

Flagi statusu: Są to flagi wskazujące, czy komunikat fizyczny należy do grupy komunikatów, czy jest segmentem komunikatu logicznego, czy też nie. W wywołaniu MQPUT lub MQPUT1 albo zwracanym przez wywołanie MQGET można określić co najmniej jeden z następujących elementów:

MMIG
Komunikat jest elementem grupy.

MFLMIG@ item: indo
Komunikat jest ostatnim logicznym komunikatem w grupie.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza element MFMIG w kopii deskryptora MQMD wysyłanej z komunikatem, ale nie zmienia ustawień tych flag w deskryptorze MQMD udostępnianym przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Poprawne jest, aby grupa składała się tylko z jednego komunikatu logicznego. W takim przypadku ustawiona jest wartość MFLMIG, ale pole MDSEQ ma wartość 1.

MSEG
Komunikat jest segmentem komunikatu logicznego.

Jeśli parametr MFSEG jest określony bez parametru MFLSEG, długość danych komunikatu aplikacji w segmencie (z wyłączeniem długości dowolnej struktury nagłówka produktu MQ , która może być obecna) musi wynosić co najmniej jeden. Jeśli długość wynosi zero, wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem przyczyny RC2253.

MFLSEG
Komunikat jest ostatnim segmentem komunikatu logicznego.

Jeśli ta opcja jest ustawiona, menedżer kolejek włącza opcję MFSEG w kopii deskryptora MQMD wysyłanej z komunikatem, ale nie zmienia ustawień tych flag w deskryptorze MQMD udostępnianym przez aplikację w wywołaniu MQPUT lub MQPUT1 .

Komunikat logiczny może składać się tylko z jednego segmentu. W takim przypadku pole MFLSEG jest ustawione, ale pole MDOFF ma wartość zero.

Jeśli określono MFLSEG, dozwolona jest zerowa długość danych komunikatu aplikacji w segmencie (z wyłączeniem długości wszystkich struktur nagłówka, które mogą być obecne).

Aplikacja musi upewnić się, że te flagi są poprawnie ustawione podczas umieszczania komunikatów. Jeśli określono PMLOGO lub określono go w poprzednim wywołaniu MQPUT dla uchwytu kolejki, ustawienia flag muszą być spójne z informacjami o grupie i segmencie zachowanymi przez menedżer kolejek dla uchwytu kolejki. Następujące warunki mają zastosowanie do kolejnych wywołań MQPUT dla uchwytu kolejki, gdy określono PMLOGO:

- Jeśli nie ma bieżącej grupy lub komunikatu logicznego, wszystkie te flagi (i ich kombinacje) są poprawne.
- Po określeniu MFMIG musi on pozostać włączony do czasu określenia MFLMIG. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2241 .
- Po określeniu elementu MFSEG musi on pozostać włączony do czasu określenia elementu MFLSEG. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2242 .

- Po określeniu MFSEG bez MFMIG, MFMIG musi pozostać wyłączony do czasu określenia MFLSEG. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2242 .

Tabela 1 przedstawia poprawne kombinacje flag i wartości używane w różnych polach.

Są to flagi wejściowe w wywołaniach MQPUT i MQPUT1 oraz flagi wyjściowe w wywołaniu MQGET. W drugim wywołaniu menedżer kolejek również echa wartości flag do pól GMGST i GMSST w MQGMO.

Opcje domyślne: Aby wskazać, że komunikat ma atrybuty domyślne, można określić następujące wartości:

BRAK MFNONE
Brak flag komunikatów (domyślne atrybuty komunikatów).

Powoduje to zablokowanie segmentacji i wskazuje, że komunikat nie znajduje się w grupie i nie jest segmentem komunikatu logicznego. Element MFNONE jest zdefiniowany w celu wspomagania dokumentacji programu. Nie jest to zamierzone, aby ta flaga była używana z innymi, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Pole MDMFL jest podzielone na podpola; szczegółowe informacje zawiera sekcja “Opcje raportu i flagi komunikatów w systemie IBM i” na stronie 1478.

Wartością początkową tego pola jest MFNONE. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDMID (24-bajtowy łańcuch bitowy)

Identyfikator komunikatu.

Jest to łańcuch bajtowy używany do odróżniania jednego komunikatu od drugiego. Ogólnie rzecz biorąc, żaden z dwóch komunikatów nie powinien mieć tego samego identyfikatora komunikatu, chociaż nie jest to niedozwolone przez menedżer kolejek. Identyfikator komunikatu jest trwałą właściwością komunikatu i jest zachowywany po restarcie menedżera kolejek. Ponieważ identyfikator komunikatu jest łańcuchem bajtowym, a nie łańcuchem znaków, identyfikator komunikatu nie jest przekształcany między zestawami znaków, gdy komunikat przepływa z jednego menedżera kolejek do innego.

W przypadku wywołań MQPUT i MQPUT1 , jeśli aplikacja określa parametr MINONE lub PMNMID, menedżer kolejek generuje unikalny identyfikator komunikatu podczas umieszczania komunikatu i umieszcza go w deskryptorze komunikatu wysłanym razem z komunikatem. Menedżer kolejek zwraca również ten identyfikator komunikatu w deskryptorze komunikatu należącym do aplikacji wysyłającej. Aplikacja może użyć tej wartości do rejestrowania informacji o konkretnych komunikatach i do odpowiadania na zapytania z innych części aplikacji.

Kod MDMID wygenerowany przez menedżera kolejek składa się z 4-bajtowego identyfikatora produktu (AMQ¬ lub CSQ¬ w kodzie ASCII lub EBCDIC, gdzie ¬ reprezentuje pojedynczy znak odstępu), po którym następuje specyficzna dla produktu implementacja unikalnego łańcucha. W pliku IBM MQ zawiera on pierwsze 12 znaków nazwy menedżera kolejek i wartość pochodzącą z zegara systemowego. Wszystkie menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się pierwszymi 12 znakami, aby identyfikatory komunikatów były unikalne. Możliwość generowania unikalnego łańcucha zależy również od tego, czy zegar systemowy nie jest zmieniany wstecz. Aby uniknąć możliwości wygenerowania przez menedżer kolejek identyfikatora komunikatu, który jest duplikatem identyfikatora wygenerowanego przez aplikację, aplikacja powinna unikać generowania identyfikatorów ze znakami początkowymi z zakresu od A do I w kodzie ASCII lub EBCDIC (X'41 'do X'49' i X'C1'do X'C9'). Jednak aplikacja nie może generować identyfikatorów z początkowymi znakami w tych zakresach.

Jeśli komunikat jest umieszczany w temacie, menedżer kolejek generuje unikalne identyfikatory komunikatów zgodnie z potrzebami dla każdego publikowanego komunikatu. Jeśli identyfikator PMNMID jest określony przez aplikację, menedżer kolejek generuje unikalny identyfikator komunikatu, który ma być zwracany po wyjściu. Jeśli wartość MINONE jest określona przez aplikację, wartość pola MDMID w deskryptorze MQMD pozostaje niezmieniona po powrocie z wywołania.

Więcej informacji na temat zachowanych publikacji zawiera opis narzędzia PMRET w sekcji PMOPT .

Jeśli komunikat jest umieszczany na liście dystrybucyjnej, menedżer kolejek w razie potrzeby generuje unikalne identyfikatory komunikatów, ale wartość pola MDMID w deskryptorze MQMD pozostaje niezmieniona po powrocie z wywołania, nawet jeśli określono opcję MINONE lub PMNMID. Jeśli aplikacja musi znać identyfikatory komunikatów wygenerowane przez menedżer kolejek, musi udostępnić rekordy MQPMR zawierające pole PRMID .

Aplikacja wysyłająca może również określić konkretną wartość dla identyfikatora komunikatu inną niż MINONE. Zatrzymuje to generowanie przez menedżer kolejek unikalnego identyfikatora komunikatu. Aplikacja, która przekazuje komunikat, może użyć tego narzędzia do propagowania identyfikatora oryginalnego komunikatu.

Menedżer kolejek sam nie używa tego pola, z wyjątkiem:

- Generuj unikalną wartość na żądanie, zgodnie z wcześniejszym opisem
- Dostarcz wartość do aplikacji, która wysyła żądanie pobrania komunikatu
- Skopiuj wartość do pola MDCID dowolnego komunikatu raportu, który jest generowany dla tego komunikatu (w zależności od opcji MDREP).

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, ustawia pole MDMID w sposób określony w polu MDREP oryginalnego komunikatu (RONMI lub ROPMI). Należy to zrobić również w przypadku aplikacji generujących komunikaty raportów.

W przypadku wywołania MQGET MDMID jest jednym z pięciu pól, za pomocą których można wybrać konkretny komunikat do pobrania z kolejki. Zwykle wywołanie MQGET zwraca następny komunikat w kolejce, ale jeśli konkretny komunikat jest wymagany, można go uzyskać, określając co najmniej jedno z pięciu kryteriów wyboru w dowolnej kombinacji. Są to następujące pola:

- MDMID
- MDCID
- MDGID
- MDSEQ
- MDOFF

Aplikacja ustawia co najmniej jedno z tych pól na wymagane wartości, a następnie ustawia odpowiednie opcje zgodności MO* w polu GMMO w MQGMO, aby wskazać, że te pola powinny być używane jako kryteria wyboru. Tylko komunikaty, które mają określone wartości w tych polach, są kandydatami do pobrania. Wartość domyślna dla pola GMMO (jeśli nie została zmieniona przez aplikację) jest zgodna zarówno z identyfikatorem komunikatu, jak i identyfikatorem korelacji.

Zwykle zwracany komunikat jest pierwszym komunikatem w kolejce, który spełnia kryteria wyboru. Jeśli jednak podano wartość GMBRWN, zwracany jest następny komunikat, który spełnia kryteria wyboru. Skanowanie tego komunikatu rozpoczyna się od komunikatu następującego po bieżącej pozycji kursora.

Uwaga: Kolejka jest skanowana sekwencyjnie w poszukiwaniu komunikatu spełniającego kryteria wyboru, dlatego czasy pobierania będą wolniejsze niż w przypadku braku kryteriów wyboru, szczególnie jeśli przed znalezniem odpowiedniego komunikatu należy przeskanować wiele komunikatów.

Więcej informacji na temat sposobu używania kryteriów wyboru w różnych sytuacjach zawiera Tabela 1 .

Określenie MINONE jako identyfikatora komunikatu ma taki sam efekt, jak nieokreślenie MOMSGI, co oznacza, że dowolny identyfikator komunikatu będzie zgodny.

To pole jest ignorowane, jeśli w parametrze **GMO** wywołania MQGET podano opcję GMMUC.

W przypadku powrotu z wywołania MQGET pole MDMID jest ustawiane na identyfikator zwróconego komunikatu (jeśli istnieje).

Można użyć następującej wartości specjalnej:

MINONE
Nie określono identyfikatora komunikatu.

Wartością długości pola jest zero binarne.

Jest to pole wejściowe/wyjściowe dla wywołań MQGET, MQPUT i MQPUT1 . Długość tego pola jest określona przez LNMID. Wartością początkową tego pola jest MINONE.

MDMT (10-cyfrowa liczba całkowita ze znakiem)

Typ wiadomości.

Wskazuje typ komunikatu. Typy komunikatów są pogrupowane w następujący sposób:

MTSFST
Najniższa wartość dla typów komunikatów zdefiniowanych przez system.

MTSLST@ item: inmenu
Najwyższa wartość dla typów komunikatów zdefiniowanych przez system.

Następujące wartości są obecnie zdefiniowane w zakresie systemowym:

MTGRM
Komunikat nie wymaga odpowiedzi.

Komunikat nie wymaga odpowiedzi.

MTRQST
Komunikat wymagający odpowiedzi.

Komunikat wymaga odpowiedzi.

Nazwa kolejki, do której ma zostać wysłana odpowiedź, musi być określona w polu MDRQ . Pole MDREP wskazuje, w jaki sposób mają zostać ustawione wartości MDMID i MDCID odpowiedzi.

MTRPLY
Odpowiedz na wcześniejszy komunikat żądania.

Komunikat jest odpowiedzią na wcześniejszy komunikat żądania (MTRQST). Komunikat powinien zostać wysłany do kolejki wskazanej w polu MDRQ komunikatu żądania. Pole MDREP żądania powinno być używane do sterowania sposobem ustawiania wartości MDMID i MDCID odpowiedzi.

Uwaga: Menedżer kolejek nie wymusza relacji żądanie-odpowiedź; jest to odpowiedzialność za aplikację.

MTRPRT
Komunikat raportu.

Komunikat jest raportowany w przypadku nieoczekiwanego lub oczekiwanego wystąpienia, zwykle związanego z innym komunikatem (na przykład odebrano komunikat żądania, który zawierał niepoprawne dane). Komunikat powinien zostać wysłany do kolejki wskazanej w polu MDRQ deskryptora oryginalnego komunikatu. Pole MDFB powinno być ustawione w taki sposób, aby wskazywały rodzaj raportu. Pole MDREP oryginalnego komunikatu może być używane do sterowania sposobem ustawiania wartości MDMID i MDCID komunikatu raportu.

Komunikaty raportów wygenerowane przez menedżera kolejek lub agenta kanału komunikatów są zawsze wysyłane do kolejki MDRQ z polami MDFB i MDCID ustawionymi zgodnie z wcześniejszym opisem.

Inne wartości z zakresu systemu mogą być zdefiniowane w przyszłych wersjach interfejsu MQI i są bez błędu akceptowane przez wywołania MQPUT i MQPUT1 .

Można również użyć wartości zdefiniowanych przez aplikację. Muszą one mieścić się w następującym zakresie:

MTAFST
Najniższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

MKALST
Najwyższa wartość dla typów komunikatów zdefiniowanych przez aplikację.

W przypadku wywołań MQPUT i MQPUT1 wartość MDMT musi być w zakresie zdefiniowanym przez system lub w zakresie zdefiniowanym przez aplikację. Jeśli nie, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2029.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest MTDGRM.

MDOFF (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego.

Jest to przesunięcie w bajtach danych w komunikacie fizycznym od początku komunikatu logicznego, którego dane stanowią część. Te dane są nazywane *segmentem*. Przesunięcie jest z zakresu od 0 do 999 999 999. Komunikat fizyczny, który nie jest segmentem komunikatu logicznego, ma przesunięcie równe zero.

To pole nie musi być ustawiane przez aplikację w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono PMLOGO.
- W wywołaniu MQGET nie określono parametru MOOFFS.

Są to zalecane sposoby używania tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja nie spełnia tych warunków lub wywołanie ma wartość MQPUT1, aplikacja musi upewnić się, że parametr MDOFF ma odpowiednią wartość.

Na wejściu wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości określonej w Tabeli 1. W danych wyjściowych wywołań MQPUT i MQPUT1 menedżer kolejek ustawia w tym polu wartość, która została wysłana razem z komunikatem.

W przypadku komunikatu raportu raportującego dla segmentu komunikatu logicznego pole MDOLN (pod warunkiem, że nie jest to wartość OLUNDF) jest używane do aktualizacji przesunięcia w informacjach o segmencie przechowywanych przez menedżer kolejek.

Na wejściu wywołania MQGET menedżer kolejek używa wartości określonej w Tabeli 1. W danych wyjściowych wywołania MQGET menedżer kolejek ustawia w tym polu wartość pobranego komunikatu.

Wartością początkową tego pola jest zero. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDOLN (10-cyfrowa liczba całkowita ze znakiem)

Długość oryginalnego komunikatu.

To pole ma znaczenie tylko w przypadku komunikatów raportu, które są segmentami. Określa on długość segmentu komunikatu, do którego odnosi się komunikat raportu; nie określa długości komunikatu logicznego, do którego należy segment, ani długości danych w komunikacie raportu.

Uwaga: Podczas generowania komunikatu raportu dla komunikatu, który jest segmentem, menedżer kolejek i agent kanału komunikatów kopiują do deskryptora MQMD dla komunikatu raportu pola MDGID, MDSEQ, MDOFFi *MDMFL*z oryginalnego komunikatu. W rezultacie komunikat raportu jest również segmentem. Zalecane jest, aby aplikacje generujące komunikaty raportów wykonowały tę samą czynność i aby zapewnić poprawne ustawienie pola MDOLN .

Zdefiniowana jest następująca wartość specjalna:

OLUNDF
Pierwotna długość komunikatu nie została zdefiniowana.

MDOLN to pole wejściowe w wywołaniach MQPUT i MQPUT1 , ale wartość udostępniona przez aplikację jest akceptowana tylko w określonych okolicznościach:

- Jeśli umieszczany komunikat jest segmentem, a także komunikatem raportu, menedżer kolejek akceptuje podaną wartość. Wartość musi być następująca:
  - Wartość większa od zera, jeśli segment nie jest ostatnim segmentem
  - Nie mniej niż zero, jeśli segment jest ostatnim segmentem

– Nie mniej niż długość danych obecnych w komunikacie

Jeśli te warunki nie są spełnione, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2252.

- Jeśli umieszczany komunikat jest segmentem, ale nie komunikatem raportu, menedżer kolejek ignoruje pole i zamiast niego używa długości danych komunikatu aplikacji.
- We wszystkich innych przypadkach menedżer kolejek ignoruje pole i używa zamiast niego wartości OLUNDF.

Jest to pole wyjściowe wywołania MQGET.

Wartością początkową tego pola jest OLUNDF. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDPAN (28-bajtowy łańcuch znaków)

Nazwa aplikacji, która umieściła komunikat.

Jest to część *kontekstu źródłowego* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Format parametru MDPAN zależy od wartości parametru MDPAT.

Jeśli to pole jest ustawiane przez menedżer kolejek (czyli dla wszystkich opcji z wyjątkiem PMSETA), jest ustawiane na wartość, która jest określana przez środowisko:

- z/OS W systemie z/OSmenedżer kolejek używa:
  - W przypadku zadania wsadowego z/OS jest to 8-znakowa nazwa zadania z karty JES JOB
  - W przypadku TSO-7-znakowy identyfikator użytkownika TSO
  - W systemie CICS: 8-znakowa zmienna applid, po której następuje 4-znakowa zmienna tranid
  - W systemie IMS: 8-znakowy identyfikator systemu IMS , po którym następuje 8-znakowa nazwa PSB
  - Dla XCF: 8-znakowa nazwa grupy XCF, po której następuje 16-znakowa nazwa podzbioru XCF
  - W przypadku komunikatu wygenerowanego przez menedżer kolejek pierwsze 28 znaków nazwy menedżera kolejek
  - W przypadku kolejkowania rozproszonego bez systemu CICS: 8-znakowa nazwa zadania inicjatora kanału, po której następuje 8-znakowa nazwa modułu umieszczającego w kolejce niedostarczonych komunikatów, po której następuje 8-znakowy identyfikator zadania.
  - Dla powiązań języka MQSeries Java przetwarzających IBM MQ for z/OS 8-znakową nazwę zadania przestrzeni adresowej utworzonej dla środowiska z/OS UNIX System Services . Zwykle jest to identyfikator użytkownika TSO z dołączonym pojedynczym znakiem liczbowym.

  Nazwa lub nazwy są dopełniane po prawej stronie odstępami, tak jak każda spacja w pozostałej części pola. Jeśli istnieje więcej niż jedna nazwa, nie ma między nimi separatora.
- Windows W systemach PC DOS i Windows menedżer kolejek używa:
  - W przypadku aplikacji CICS jest to nazwa transakcji CICS
  - W przypadku aplikacji innej niż aplikacjaCICS , 28 znaków po prawej stronie pełnej nazwy pliku wykonywalnego
- IBM i W systemie IBM imenedżer kolejek używa pełnej nazwy zadania.
- Linux AIX W systemie AIX and Linuxmenedżer kolejek używa:
  - W przypadku aplikacji CICS jest to nazwa transakcji CICS
  - W przypadku aplikacji innej niżCICS -14 skrajnie prawych znaków pełnej nazwy pliku wykonywalnego, jeśli jest ona dostępna dla menedżera kolejek, w przeciwnym razie odstępy (na przykład w systemie AIX)

- W systemie VSE/ESAmenedżer kolejek używa 8-znakowego identyfikatora applid, po którym następuje 4-znakowy identyfikator tranid.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono wartość PMSETA. Wszelkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Znak o kodzie zero i wszystkie następujące po nim znaki są przekształcane przez menedżera kolejek w odstępy. Jeśli parametr PMSETA nie jest określony, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez LNPAN. Wartością początkową tego pola jest 28 znaków odstępu.

MDPAT (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji, która umieściła komunikat.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

MDPAT może mieć jeden z następujących typów standardowych. Typy zdefiniowane przez użytkownika mogą być również używane, ale powinny być ograniczone do wartości z zakresu od ATUFST do ATULST.

ATAIX,
Aplikacja AIX (taka sama jak ATUNIX).

ATBRKR
Broker.

ATCICS,
CICS .

ATCICB
CICS bridge.

ATVSE,
CICS/VSE .

ATDOS,
Aplikacja IBM MQ MQI client w systemie DOS na komputerze PC.

ATDQM,
Agent rozproszonego menedżera kolejek.

ATGUAR
Aplikacja Tandem Guardian (taka sama wartość jak ATNSK).

ATIMS,
Aplikacja IMS .

ATIMSB
Most IMS .

ATJAVA
Java.

ATMVS,
Aplikacja MVS lub TSO (taka sama wartość jak ATZOS).

UWAGA
Lotus Notes Aplikacja agenta.

ATNSK
Tandem NonStop aplikacja jądra.

AT390
Aplikacja OS/390 (taka sama jak ATZOS).

AT400
Aplikacja IBM i .

ATQM,
menedżerze kolejek.

System ATUNIX
Aplikacja UNIX .

ATVOS (system operacyjny)
Aplikacja Stratus VOS.

ATWIN
16-bitowa aplikacja Windows .

ATWINT.PL
32-bitowa aplikacja Windows .

ATXCF,
XCF.

ATZOS
Aplikacja z/OS .

ATDEF,
Domyślny typ aplikacji.

Jest to domyślny typ aplikacji dla platformy, na której działa aplikacja.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

ATUNK
Nieznany typ aplikacji.

Ta wartość może być używana do wskazania, że typ aplikacji jest nieznany, nawet jeśli istnieją inne informacje o kontekście.

ATUFST
Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

ATULST
Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Może również wystąpić następująca wartość specjalna:

ATNCON
Brak informacji o kontekście w komunikacie.

Ta wartość jest ustawiana przez menedżer kolejek, gdy komunikat jest umieszczany bez kontekstu (oznacza to, że podano opcję kontekstu PMNOC).

Po pobraniu komunikatu produkt MDPAT może zostać przetestowany pod kątem tej wartości w celu określenia, czy komunikat ma kontekst (zaleca się, aby aplikacja używająca narzędzia PMSETA nigdy nie ustawiała właściwości MDPAT na wartość ATNCON, jeśli jakiekolwiek inne pola kontekstu nie są puste).

ATSIB
Wskazuje, że komunikat pochodzi z innego produktu przesyłania komunikatów IBM MQ i został odebrany przez most magistrali integracji usług (Service Integration Bus).

Gdy menedżer kolejek generuje te informacje w wyniku umieszczenia w aplikacji, pole jest ustawiane na wartość, która jest określana przez środowisko.

IBM i Należy zauważyć, że w systemie IBM ipole jest ustawione na wartość AT400; menedżer kolejek nigdy nie używa programu ATCICS w systemie IBM i.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono wartość PMSETA. Jeśli parametr PMSETA nie jest określony, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość MDPAT , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość

parametru MDPAT , która jest przechowywana razem z komunikatem, jeśli zostanie zachowana (więcej informacji na temat zachowanych publikacji zawiera opis PMRET), ale nie jest używana jako parametr MDPAT , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ we wszystkich wysłanych do nich publikacjach podano wartość przesłaniającą parametr MDPAT . Jeśli komunikat nie ma kontekstu, pole jest ustawione na wartość ATNCON.

Jest to pole wyjściowe wywołania MQGET. Wartością początkową tego pola jest ATNCON.

MDPD (8-bajtowy łańcuch znaków)

Data umieszczenia komunikatu.

Jest to część *kontekstu źródłowego* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Format używany dla daty wygenerowania tego pola przez menedżer kolejek to:

- RRRRMMDD

gdzie znaki reprezentują:

rrrr
rok (cztery cyfry)

MM
miesiąc roku (od 01 do 12)

DD
dzień miesiąca (od 01 do 31)

Czas Greenwich (Greenwich Mean Time-GMT) jest używany w polach MDPD i MDPT , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, datą jest data umieszczenia komunikatu, a nie data zatwierdzenia jednostki pracy.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono wartość PMSETA. Zawartość pola nie jest sprawdzana przez menedżer kolejek, z wyjątkiem tego, że wszystkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Znak o kodzie zero i wszystkie następujące po nim znaki są przekształcane przez menedżera kolejek w odstępy. Jeśli parametr PMSETA nie jest określony, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość MDPD , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru MDPD , która jest przechowywana razem z komunikatem, jeśli zostanie zachowana (więcej informacji na temat zachowanych publikacji zawiera opis PMRET), ale nie jest używana jako parametr MDPD , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ we wszystkich wysłanych do nich publikacjach podano wartość przesłaniającą parametr MDPD . Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez LNPDAT. Wartość początkowa tego pola to 8 znaków odstępu.

MDPER (10-cyfrowa liczba całkowita ze znakiem)

Trwałość komunikatu.

Wskazuje, czy komunikat przetrwa awarie systemu i restarty menedżera kolejek. W przypadku wywołań MQPUT i MQPUT1 wartością musi być jedna z następujących wartości:

PEPER,
Komunikat jest trwały.

Oznacza to, że komunikat przetrwa awarie systemu i restarty menedżera kolejek. Po umieszczeniu komunikatu i zatwierdzeniu jednostki pracy puttera (jeśli komunikat jest umieszczany jako część jednostki pracy), komunikat jest zachowywany w pamięci dyskowej. Pozostaje tam do momentu

usunięcia komunikatu z kolejki i zatwierdzenia jednostki pracy procedury pobierającej (jeśli komunikat jest pobierany jako część jednostki pracy).

Gdy komunikat trwały jest wysyłany do kolejki zdalnej, mechanizm przechowywania i przekazywania jest używany do przechowywania komunikatu w każdym menedżerze kolejek na trasie do miejsca docelowego, dopóki wiadomo, że komunikat dotarł do następnego menedżera kolejek.

Nie można umieścić trwałych komunikatów w:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane, w których poziom struktury narzędzia CF jest mniejszy niż trzy lub struktura narzędzia CF nie jest odtwarzalna.

Komunikaty trwałe mogą być umieszczane w trwałych kolejkach dynamicznych, predefiniowanych kolejkach i kolejkach współużytkowanych, w których poziom struktury narzędzia CF wynosi 3, a narzędzie CF jest odtwarzalne.

PENPER
Komunikat nie jest trwały.

Oznacza to, że komunikat zwykle nie przetrwa awarii systemu ani restartów menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartowania menedżera kolejek w pamięci dyskowej zostanie znaleziona nienaruszona kopia komunikatu.

W przypadku kolejek współużytkowanych komunikaty nietrwałe *pozostają* po restarcie menedżerów kolejek w grupie współużytkowania kolejek, ale nie są zachowywane awarie narzędzia CF używanego do przechowywania komunikatów we współużytkowanych kolejkach.

PEQDEF,
Komunikat ma domyślną trwałość.

- Jeśli kolejka jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu **DefPersistence** zdefiniowanego w docelowym menedżerze kolejek, który jest właścicielem konkretnej instancji kolejki, w której został umieszczony komunikat. Zwykle wszystkie instancje kolejki klastra mają taką samą wartość atrybutu **DefPersistence** , chociaż nie jest to wymagane.

  Wartość **DefPersistence** jest kopiowana do pola *MDPER* , gdy komunikat jest umieszczany w kolejce docelowej. Jeśli parametr **DefPersistence** zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone w kolejce.
- Jeśli kolejka nie jest kolejką klastra, trwałość komunikatu jest pobierana z atrybutu **DefPersistence** zdefiniowanego w lokalnym menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

  Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w pierwszej definicji ścieżki. Może to być:

  – Kolejka aliasowa
  – Kolejka lokalna
  – Lokalna definicja kolejki zdalnej
  – Alias menedżera kolejek
  – Kolejka transmisji (na przykład `DefXmitQName`)

  Wartość **DefPersistence** jest kopiowana do pola `MDPER` podczas umieszczania komunikatu. Jeśli plik **DefPersistence** zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone.

W tej samej kolejce mogą istnieć zarówno komunikaty trwałe, jak i nietrwałe.

Podczas odpowiadania na komunikat aplikacje powinny zwykle używać trwałości komunikatu żądania dla komunikatu odpowiedzi.

W przypadku wywołania MQGET zwracana jest wartość PEPER lub PENPER.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest PEQDEF.

MDPRI (10-cyfrowa liczba całkowita ze znakiem)

Priorytet komunikatu.

W przypadku wywołań MQPUT i MQPUT1 wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następującej wartości specjalnej:

PRQDEF,
Domyślny priorytet kolejki.

- Jeśli kolejka jest kolejką klastra, priorytet komunikatu jest pobierany z atrybutu **DefPriority** zdefiniowanego w docelowym menedżerze kolejek, który jest właścicielem konkretnej instancji kolejki, w której umieszczono komunikat. Zwykle wszystkie instancje kolejki klastra mają taką samą wartość atrybutu **DefPriority** , chociaż nie jest to wymagane.

  Wartość **DefPriority** jest kopiowana do pola MDPRI , gdy komunikat jest umieszczany w kolejce docelowej. Jeśli parametr **DefPriority** zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone w kolejce.
- Jeśli kolejka nie jest kolejką klastra, priorytet komunikatu jest pobierany z atrybutu **DefPriority** zdefiniowanego w lokalnym menedżerze kolejek, nawet jeśli docelowy menedżer kolejek jest zdalny.

  Jeśli w ścieżce rozstrzygania nazwy kolejki znajduje się więcej niż jedna definicja, priorytet domyślny jest pobierany z wartości tego atrybutu w pierwszej definicji ścieżki. Może to być:

  – Kolejka aliasowa
  – Kolejka lokalna
  – Lokalna definicja kolejki zdalnej
  – Alias menedżera kolejek
  – Kolejka transmisji (na przykład DefXmitQName)

  Wartość **DefPriority** jest kopiowana do pola MDPRI podczas umieszczania komunikatu. Jeśli plik **DefPriority** zostanie później zmieniony, nie będzie to miało wpływu na komunikaty, które zostały już umieszczone.

Wartość zwracana przez wywołanie MQGET jest zawsze większa lub równa zero; wartość PRQDEF nigdy nie jest zwracana.

Jeśli komunikat jest umieszczany z priorytetem większym niż maksimum obsługiwane przez lokalny menedżer kolejek (to maksimum jest nadawane przez atrybut **MaxPriority** menedżera kolejek), komunikat jest akceptowany przez menedżer kolejek, ale umieszczany w kolejce z maksymalnym priorytetem menedżera kolejek; wywołanie MQPUT lub MQPUT1 kończy się komunikatem CCWARN i kodem przyczyny RC2049. Jednak pole MDPRI zachowuje wartość określoną przez aplikację, która umieściła komunikat.

Podczas odpowiadania na komunikat aplikacje powinny zwykle używać dla komunikatu odpowiedzi priorytetu komunikatu żądania. W innych sytuacjach określenie parametru PRQDEF umożliwia strojenie priorytetów bez zmiany aplikacji.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest PRQDEF.

MDPT (8-bajtowy łańcuch znaków)

Czas umieszczenia komunikatu.

Jest to część **kontekstu źródłowego** komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Format używany dla czasu wygenerowania tego pola przez menedżer kolejek to:

- GGMMSSTH

gdzie znaki reprezentują (w kolejności):

GG
godzin (od 00 do 23)

MM
minuty (od 00 do 59)

SS
sekundy (od 00 do 59; patrz uwaga)

T
dziesiąte części sekundy (od 0 do 9)

H
setne sekundy (od 0 do 9)

Uwaga: Jeśli zegar systemowy jest zsynchronizowany z bardzo dokładnym standardem czasu, w rzadkich przypadkach możliwe jest zwrócenie wartości 60 lub 61 dla sekund w produkcie MDPT. Dzieje się tak, gdy sekundy przestępne są wstawiane do globalnego standardu czasu.

Czas Greenwich (Greenwich Mean Time-GMT) jest używany w polach MDPD i MDPT , pod warunkiem, że zegar systemowy jest dokładnie ustawiony na czas GMT.

Jeśli komunikat został umieszczony jako część jednostki pracy, jest to czas umieszczenia komunikatu, a nie czas zatwierdzenia jednostki pracy.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono wartość PMSETA. Zawartość pola nie jest sprawdzana przez menedżer kolejek, z wyjątkiem tego, że wszystkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Znak o kodzie zero i wszystkie następujące po nim znaki są przekształcane przez menedżera kolejek w odstępy. Jeśli parametr PMSETA nie jest określony, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość MDPT , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru MDPT , która jest przechowywana razem z komunikatem, jeśli zostanie zachowana (więcej informacji na temat zachowanych publikacji zawiera opis PMRET), ale nie jest używana jako parametr MDPT , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ we wszystkich wysłanych do nich publikacjach podano wartość przesłaniającą parametr MDPT . Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez LNPTIM. Wartość początkowa tego pola to 8 znaków odstępu.

MDREP (10-cyfrowa liczba całkowita ze znakiem)

Opcje komunikatów raportu.

Komunikat raportu to komunikat o innym komunikacie, używany do informowania aplikacji o oczekiwanych lub nieoczekiwanych zdarzeniach, które odnoszą się do oryginalnego komunikatu. Pole MDREP umożliwia aplikacji wysyłającej oryginalny komunikat w celu określenia, które komunikaty raportu są wymagane, czy dane komunikatu aplikacji mają być w nich uwzględnione, a także (zarówno w przypadku raportów, jak i odpowiedzi) w jaki sposób mają zostać ustawione identyfikatory komunikatu i korelacji w komunikacie raportu lub odpowiedzi. Można zażądać dowolnego lub wszystkich (lub żadnego) następujących typów komunikatów raportu:

- Wyjątek
- Termin ważności
- Potwierdź po przybyciu (COA)
- Potwierdź przy dostawie (COD)
- Powiadomienie o działaniu pozytywnym (PAN)
- Powiadomienie o działaniu negatywnym (NAN)

Jeśli wymagany jest więcej niż jeden typ komunikatu raportu lub wymagane są inne opcje raportu, wartości mogą zostać dodane do siebie (nie należy dodawać tej samej stałej więcej niż raz).

Aplikacja, która odbiera komunikat raportu, może określić przyczynę wygenerowania raportu, sprawdzając pole MDFB w strukturze MQMD. Więcej szczegółów zawiera pole MDFB .

Użycie opcji raportu podczas umieszczania komunikatu w temacie może spowodować wygenerowanie i wysłanie do aplikacji jednego lub wielu komunikatów raportu. Jest to spowodowane tym, że komunikat o publikacji może zostać wysłany do zero, jednej lub wielu aplikacji subskrybujących.

Opcje wyjątku: Aby zażądać komunikatu raportu wyjątku, można określić jedną z następujących opcji.

ROAKTYWNOŚĆ

Wymagane raporty aktywności

Ta opcja raportu umożliwia wygenerowanie raportu aktywności za każdym razem, gdy komunikat z tym zestawem opcji raportu jest przetwarzany przez aplikacje pomocnicze.

Komunikaty z tym zestawem opcji raportu muszą zostać zaakceptowane przez dowolny menedżer kolejek, nawet jeśli nie rozumie on tej opcji. Umożliwia to ustawienie opcji raportu dla dowolnego komunikatu użytkownika, nawet jeśli są one przetwarzane przez poprzednie menedżery kolejek. W tym celu opcja raportu jest umieszczana w podpolu ROAUM.

Jeśli proces (menedżer kolejek lub proces użytkownika) wykonuje działanie na komunikacie z ustawioną wartością ROACT, może wygenerować i umieścić raport działań.

Opcja raportu aktywności umożliwia śledzenie trasy dowolnego komunikatu w sieci menedżera kolejek. Opcja raportu może być określona dla każdego komunikatu bieżącego użytkownika i może natychmiast rozpocząć obliczanie trasy komunikatu przez sieć. Jeśli aplikacja generująca komunikat nie może włączyć generowania raportu aktywności, można go włączyć za pomocą wyjścia przecięcia funkcji API udostępnianego przez administratorów menedżera kolejek.

Do raportów z aktywności ma zastosowanie kilka warunków:

1. Trasa będzie mniej szczegółowa, jeśli w sieci będzie mniej menedżerów kolejek, które mogą generować raporty działań.
2. Raporty z działalności mogą nie być łatwo "zamawiane" w celu określenia podjętej trasy.
3. Raporty aktywności mogą nie być w stanie znaleźć trasy do żądanego miejsca docelowego.

ROEXC

Wymagane są raporty o wyjątkach.

Ten typ raportu może zostać wygenerowany przez agenta kanału komunikatów, gdy komunikat jest wysyłany do innego menedżera kolejek i nie może zostać dostarczony do określonej kolejki docelowej. Na przykład kolejka docelowa lub pośrednia kolejka transmisji może być pełna lub komunikat może być zbyt duży dla kolejki.

Generowanie komunikatu raportu o wyjątku zależy od trwałości oryginalnego komunikatu i szybkości kanału komunikatów (normalnego lub szybkiego), przez który przechodzi oryginalny komunikat:

- Dla wszystkich trwałych komunikatów oraz dla nietrwałych komunikatów przechodujących przez normalne kanały komunikatów raport o wyjątkach jest generowany tylko wtedy, gdy działanie określone przez aplikację wysyłającą dla warunku błędu może zostać zakończone pomyślnie. Aplikacja wysyłająca może określić jedną z następujących czynności w celu sterowania rozdysponowaniem oryginalnego komunikatu w przypadku wystąpienia warunku błędu:
  - RODLQ (powoduje umieszczenie oryginalnego komunikatu w kolejce niedostarczonych komunikatów).
  - RODISC (spowoduje to usunięcie oryginalnego komunikatu).

  Jeśli działanie określone przez aplikację wysyłającą nie może zostać zakończone pomyślnie, oryginalny komunikat pozostaje w kolejce transmisji i nie jest generowany żaden komunikat raportu o wyjątku.

- W przypadku nietrwałych komunikatów przechodzącego przez kanały szybkich komunikatów oryginalny komunikat jest usuwany z kolejki transmisji i generowany jest raport o wyjątku, nawet jeśli określone działanie dla warunku błędu nie może zostać zakończone pomyślnie. Na przykład, jeśli określono RODLQ, ale nie można umieścić oryginalnego komunikatu w kolejce niedostarczonych komunikatów, ponieważ (powiedzmy) kolejka jest pełna, zostanie wygenerowany komunikat raportu o wyjątku i oryginalny komunikat zostanie odrzucony.

  Więcej informacji na temat normalnych i szybkich kanałów komunikatów zawiera sekcja Trwałość komunikatu .

Raport o wyjątku nie jest generowany, jeśli aplikacja, która umieściła oryginalny komunikat, może zostać powiadomiona synchronicznie o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 .

Aplikacje mogą również wysyłać raporty o wyjątkach w celu wskazania, że odebrany komunikat nie może zostać przetworzony (na przykład dlatego, że jest to transakcja debetowa, która spowodowałaby przekroczenie limitu kredytowego konta).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy podawać więcej niż jednej wartości ROEXC, ROEXCD i ROEXCF.

ROEXCD
Raporty wyjątków z wymaganymi danymi.

Jest taka sama jak ROEXC, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest uwzględnianych w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości ROEXC, ROEXCD i ROEXCF.

ROEXCF,
Raporty wyjątków z wymaganymi pełnymi danymi.

Jest to takie samo jak ROEXC, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości ROEXC, ROEXCD i ROEXCF.

Opcje utraty ważności: Aby zażądać komunikatu raportu o utracie ważności, można określić jedną z następujących opcji.

ROEXP
Wymagane są raporty o utracie ważności.

Ten typ raportu jest generowany przez menedżer kolejek, jeśli komunikat zostanie odrzucony przed dostarczeniem do aplikacji, ponieważ upłynął jego czas ważności (patrz pole MDEXP). Jeśli ta opcja nie jest ustawiona, komunikat raportu nie jest generowany, jeśli komunikat zostanie odrzucony z tego powodu (nawet jeśli określono jedną z opcji ROEXC*).

Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Nie należy podawać więcej niż jednej wartości ROEXP, ROEXPD i ROEXPF.

ROEXPD
Raporty o utracie ważności z wymaganymi danymi.

Jest to takie samo jak ROEXP, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest uwzględnianych w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości ROEXP, ROEXPD i ROEXPF.

ROEXPF
Raporty o utracie ważności z wymaganymi pełnymi danymi.

Jest to takie samo jak ROEXP, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są dołączane do komunikatu raportu.

Nie należy podawać więcej niż jednej wartości ROEXP, ROEXPD i ROEXPF.

Opcje potwierdzania przy odbiorze: Aby zażądać komunikatu raportu potwierdzenia przy odbiorze, można podać jedną z następujących opcji.

ROKOA
Wymagane są raporty potwierdzenia przy odbiorze.

Ten typ raportu jest generowany przez menedżera kolejek, który jest właścicielem kolejki docelowej, gdy komunikat jest umieszczany w kolejce docelowej. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest umieszczany jako część jednostki pracy, a kolejka docelowa jest kolejką lokalną, komunikat raportu COA wygenerowany przez menedżer kolejek staje się dostępny do pobrania tylko wtedy, gdy jednostka pracy została zatwierdzona.

Raport COA nie jest generowany, jeśli pole MDFMT w deskryptorze komunikatu to FMXQH lub FMDLH. Zapobiega to generowaniu raportu COA, jeśli komunikat jest umieszczany w kolejce transmisji lub nie można go dostarczyć i umieścić w kolejce niedostarczonych komunikatów.

Nie należy podawać więcej niż jednej wartości ROCOA, ROCOAD i ROCOAF.

ROKOD
Raporty potwierdzenia przy odbiorze z wymaganymi danymi.

Jest taka sama jak ROCOA, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest dołączanych do komunikatu raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości ROCOA, ROCOAD i ROCOAF.

ROKOF
Raporty potwierdzenia przy odbiorze z wymaganymi pełnymi danymi.

Jest to takie samo jak ROCOA, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są uwzględniane w komunikacie raportu.

Nie należy podawać więcej niż jednej wartości ROCOA, ROCOAD i ROCOAF.

Opcje odrzucenia i utraty ważności: Można określić następującą opcję, aby ustawić czas utraty ważności i flagę odrzucenia dla komunikatów raportu.

ROPDAE
Ustaw czas utraty ważności komunikatu raportu i flagę odrzucenia.

Ta opcja zapewnia, że komunikaty raportu i komunikaty odpowiedzi dziedziczą czas utraty ważności i flagę odrzucenia (niezależnie od tego, czy mają zostać usunięte, czy nie) z oryginalnych komunikatów. Jeśli ta opcja jest ustawiona, komunikaty raportu i odpowiedzi:

1. Dziedzicz flagę RODISC (jeśli została ustawiona).
2. Dziedzicz pozostały czas utraty ważności komunikatu, jeśli komunikat nie jest raportem utraty ważności. Jeśli komunikat jest raportem o utracie ważności, czas utraty ważności jest ustawiany na 60 sekund.

Jeśli ta opcja jest ustawiona, mają zastosowanie następujące warunki:

Uwaga:

1. Komunikaty raportów i odpowiedzi są generowane z flagą odrzucenia i wartością utraty ważności i nie mogą pozostawać w systemie.
2. Komunikaty trasy śledzenia nie mogą dotrzeć do kolejek docelowych w menedżerach kolejek, które nie obsługują trasy śledzenia.

3. Kolejki nie mogą być wypełniane raportami, których nie można dostarczyć, jeśli łącza komunikacyjne są zerwane.
4. Odpowiedzi serwera komend dziedziczą pozostały czas ważności żądania.

Opcje potwierdzania przy dostarczeniu: Aby zażądać komunikatu raportu potwierdzania przy dostarczeniu, można określić jedną z następujących opcji.

RZT
Wymagane są raporty potwierdzania przy dostarczeniu.

Ten typ raportu jest generowany przez menedżer kolejek, gdy aplikacja pobiera komunikat z kolejki docelowej w sposób powodujący usunięcie komunikatu z kolejki. Dane komunikatu z oryginalnego komunikatu nie są dołączane do komunikatu raportu.

Jeśli komunikat jest wczytywany jako część jednostki pracy, komunikat raportu jest generowany w ramach tej samej jednostki pracy, dzięki czemu raport nie jest dostępny do momentu zatwierdzenia jednostki pracy. Jeśli jednostka pracy zostanie wycofana, raport nie zostanie wysłany.

Raport COD nie jest generowany, jeśli pole MDFMT w deskryptorze komunikatu ma wartość FMDLH. Zapobiega to generowaniu raportu COD, jeśli nie można dostarczyć komunikatu i jest on umieszczany w kolejce niedostarczonych komunikatów.

Wartość ROCOD jest niepoprawna, jeśli kolejka docelowa jest kolejką XCF.

Nie należy podawać więcej niż jednej wartości ROCOD, ROCODD i ROCODF.

RODOD
Raporty potwierdzenia dostarczenia z wymaganymi danymi.

Jest taka sama jak wartość ROCOD, z tą różnicą, że pierwsze 100 bajtów danych komunikatu aplikacji z oryginalnego komunikatu jest uwzględnianych w komunikacie raportu. Jeśli oryginalny komunikat zawiera co najmniej jedną strukturę nagłówka MQ , oprócz 100 bajtów danych aplikacji są one uwzględniane w komunikacie raportu.

Jeśli w wywołaniu MQGET dla oryginalnego komunikatu określono GMATM, a pobrany komunikat jest obcięty, ilość danych komunikatu aplikacji umieszczonych w komunikacie raportu wynosi co najmniej:

- Długość oryginalnego komunikatu
- 100 bajtów.

Wartość ROCODD jest niepoprawna, jeśli kolejka docelowa jest kolejką XCF.

Nie należy podawać więcej niż jednej wartości ROCOD, ROCODD i ROCODF.

ROCODF
Wymagane są raporty potwierdzenia dostarczenia z pełnymi danymi.

Jest taka sama jak opcja ROCOD, z tą różnicą, że wszystkie dane komunikatu aplikacji z oryginalnego komunikatu są uwzględniane w komunikacie raportu.

Parametr ROCODF jest niepoprawny, jeśli kolejka docelowa jest kolejką XCF.

Nie należy podawać więcej niż jednej wartości ROCOD, ROCODD i ROCODF.

Opcje powiadomienia o działaniu: Można określić jedną lub obie z następujących opcji, aby zażądać od aplikacji odbierającej wysłania komunikatu raportu o działaniu pozytywnym lub negatywnym.

ROPAN
Wymagane są raporty powiadomień o działaniach pozytywnych.

Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Wskazuje, że działanie żądane w komunikacie zostało wykonane pomyślnie. Aplikacja generująca raport określa, czy do raportu mają zostać dołączone jakiekolwiek dane.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. Do obowiązków aplikacji pobierającej należy generowanie raportu, jeśli jest to konieczne.

RONAN
Wymagane są raporty powiadomień o działaniach negatywnych.

Ten typ raportu jest generowany przez aplikację, która pobiera komunikat i działa na nim. Oznacza to, że działanie żądane w komunikacie nie zostało wykonane pomyślnie. Aplikacja generująca raport określa, czy do raportu mają zostać dołączone jakiekolwiek dane. Na przykład może być pożądane dołączenie pewnych danych wskazujących, dlaczego żądanie nie mogło zostać wykonane.

Oprócz przekazania tego żądania do aplikacji pobierającej komunikat, menedżer kolejek nie podejmuje żadnych działań w oparciu o tę opcję. Do obowiązków aplikacji pobierającej należy generowanie raportu, jeśli jest to konieczne.

Określenie, które warunki odpowiadają działaniom pozytywnym i które odpowiadają działaniom negatywnym, należy do zakresu odpowiedzialności wniosku. Jeśli jednak żądanie zostało wykonane tylko częściowo, zaleca się wygenerowanie raportu NAN zamiast raportu PAN na żądanie. Zaleca się również, aby każdy możliwy warunek odpowiadał albo działaniu pozytywnemu, albo negatywnemu, ale nie obu jednocześnie.

Opcje identyfikatora komunikatu: Można określić jedną z następujących opcji, aby sterować sposobem ustawienia `MDMID` komunikatu raportu (lub komunikatu odpowiedzi).

RONMI
Nowy identyfikator komunikatu.

Jest to działanie domyślne i wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, dla raportu lub komunikatu odpowiedzi zostanie wygenerowany nowy element `MDMID` .

ROPMI
Przekaż identyfikator komunikatu.

Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, plik `MDMID` tego komunikatu zostanie skopiowany do pliku `MDMID` komunikatu raportu lub odpowiedzi.

Wartość `MsgId` komunikatu publikacji będzie różna dla każdego subskrybenta, który otrzymuje kopię publikacji, i dlatego wartość `MsgId` skopiowana do komunikatu raportu lub odpowiedzi będzie różna dla każdego subskrybenta.

Jeśli ta opcja nie zostanie podana, przyjmowany jest RONMI.

Opcje identyfikatora korelacji: Można określić jedną z następujących opcji, aby sterować sposobem ustawiania `MDCID` komunikatu raportu (lub komunikatu odpowiedzi).

ROCMTC,
Skopiuj identyfikator komunikatu do identyfikatora korelacji.

Jest to działanie domyślne i wskazuje, że jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, plik `MDMID` tego komunikatu zostanie skopiowany do pliku `MDCID` komunikatu raportu lub odpowiedzi.

Wartość `MsgId` komunikatu publikacji będzie inna dla każdego subskrybenta, który otrzymuje kopię publikacji i dlatego wartość `MsgId` skopiowana do pliku `CorrelId` komunikatu raportu lub odpowiedzi będzie inna dla każdego z nich.

ROPCI
Przekaż identyfikator korelacji.

Jeśli w wyniku tego komunikatu zostanie wygenerowany raport lub odpowiedź, plik `MDCID` tego komunikatu zostanie skopiowany do pliku `MDCID` komunikatu raportu lub odpowiedzi.

Parametr MDCID komunikatu publikacji będzie specyficzny dla subskrybenta, chyba że używa on opcji SOSCID i ustawia pole SCDIC w pliku MQSD na wartość CINONE. Dlatego możliwe jest, że MDCID skopiowany do MDCID raportu lub komunikatu odpowiedzi będzie inny dla każdego z nich.

Jeśli ta opcja nie zostanie podana, przyjmowany jest ROCMTC.

Zaleca się, aby serwery odpowiadające na żądania lub generujące komunikaty raportów sprawdzały, czy opcje ROPMI lub ROPCI zostały ustawione w oryginalnym komunikacie. Jeśli tak, serwery powinny wykonać opisane działanie dla tych opcji. Jeśli żaden z nich nie jest ustawiony, serwery powinny wykonać odpowiednie działanie domyślne.

: Można określić jedną z następujących opcji, aby sterować umieszczaniem oryginalnego komunikatu, gdy nie może on zostać dostarczony do kolejki docelowej. Te opcje mają zastosowanie tylko do sytuacji, które spowodowałyby wygenerowanie komunikatu raportu o wyjątku, gdyby aplikacja wysyłająca zażądała takiego komunikatu. Aplikacja może ustawić opcje rozporządzania niezależnie od żądań raportów o wyjątkach.

RODLQ,
Umieść komunikat w kolejce niedostarczonych komunikatów.

Jest to działanie domyślne, które wskazuje, że komunikat powinien zostać umieszczony w kolejce niedostarczonych komunikatów, jeśli nie można dostarczyć komunikatu do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieściła oryginalny komunikat, nie może być synchronicznie powiadamiana o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 . Jeśli nadawca zażądał komunikatu o wyjątku, zostanie on wygenerowany.
- Gdy aplikacja, która wstawiła oryginalny komunikat, umieszczała go w temacie.

Jeśli nadawca zażądał raportu o wyjątku, zostanie on wygenerowany.

RODISC
Odrzuć komunikat.

Oznacza to, że komunikat powinien zostać odrzucony, jeśli nie może zostać dostarczony do kolejki docelowej. Dzieje się tak w następujących sytuacjach:

- Jeśli aplikacja, która umieściła oryginalny komunikat, nie może być synchronicznie powiadamiana o problemie za pomocą kodu przyczyny zwróconego przez wywołanie MQPUT lub MQPUT1 . Jeśli nadawca zażądał komunikatu o wyjątku, zostanie on wygenerowany.
- Gdy aplikacja, która wstawiła oryginalny komunikat, umieszczała go w temacie.

Jeśli nadawca zażądał raportu o wyjątku, zostanie on wygenerowany.

Jeśli konieczne jest zwrócenie oryginalnego komunikatu do nadawcy bez umieszczania oryginalnego komunikatu w kolejce niedostarczonych komunikatów, nadawca powinien określić parametr RODISC z wartością ROEXCF.

Opcja domyślna: Jeśli nie są wymagane żadne opcje raportu, można określić następujące opcje:

BRAK
Raporty nie są wymagane.

Ta wartość może być używana do wskazania, że nie określono żadnych innych opcji. RONONE jest zdefiniowane w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Informacje ogólne:

1. Wszystkie wymagane typy raportów muszą być specjalnie żądane przez aplikację wysyłającą oryginalny komunikat. Na przykład, jeśli zażądano raportu COA, ale raport o wyjątku nie został wygenerowany, raport COA jest generowany, gdy komunikat jest umieszczany w kolejce docelowej, ale nie jest generowany raport o wyjątku, jeśli kolejka docelowa jest pełna w momencie nadejścia komunikatu. Jeśli nie ustawiono opcji MDREP , nie są generowane żadne komunikaty raportu przez menedżer kolejek ani agent kanału komunikatów (MCA).

Niektóre opcje raportu mogą być określone nawet wtedy, gdy lokalny menedżer kolejek ich nie rozpoznaje. Jest to przydatne, gdy opcja ma być przetwarzana przez docelowy menedżer kolejek. Więcej informacji na ten temat zawiera sekcja "Opcje raportu i flagi komunikatów w systemie IBM i" na stronie 1478.

Jeśli żądany jest komunikat raportu, nazwa kolejki, do której ma zostać wysłany raport, musi być określona w polu MDRQ . Po odebraniu komunikatu raportu rodzaj raportu można określić, sprawdzając pole MDFB w deskryptorze komunikatu.

2. Jeśli menedżer kolejek lub agent MCA generujący komunikat raportu nie może umieścić komunikatu raportu w kolejce odpowiedzi (na przykład z powodu zapełnienia kolejki odpowiedzi lub kolejki transmisji), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów. Jeśli to również nie powiedzie się lub nie ma kolejki niedostarczonych komunikatów, podjęte działanie zależy od typu komunikatu raportu:

   - Jeśli komunikat raportu jest raportem o wyjątku, komunikat, który spowodował wygenerowanie raportu o wyjątku, pozostaje w kolejce transmisji. Dzięki temu komunikat nie zostanie utracony.
   - W przypadku wszystkich innych typów raportów komunikat raportu jest odrzucany i przetwarzanie jest kontynuowane normalnie. Dzieje się tak, ponieważ oryginalny komunikat został już dostarczony bezpiecznie (dla komunikatów raportu COA lub COD) lub nie jest już interesujący (dla komunikatu raportu o utracie ważności).

   Po pomyślnym umieszczeniu komunikatu raportu w kolejce (kolejka docelowa lub pośrednia kolejka transmisji) komunikat nie podlega już specjalnemu przetwarzaniu; jest traktowany tak samo, jak każdy inny komunikat.
3. Po wygenerowaniu raportu otwierana jest kolejka MDRQ i umieszczany jest komunikat raportu z uprawnieniami MDUID w deskryptorze MQMD komunikatu będącego przyczyną raportu, z wyjątkiem następujących przypadków:

   - Raporty o wyjątkach generowane przez odbierający agent MCA są umieszczane z uprawnieniami używanymi przez agent MCA podczas próby umieszczenia komunikatu powodującego wygenerowanie raportu. Atrybut kanału CDPA określa używany identyfikator użytkownika.
   - Raporty COA wygenerowane przez menedżera kolejek są umieszczane z uprawnieniami używanymi podczas umieszczania komunikatu powodującego umieszczenie raportu w menedżerze kolejek generującym raport. Jeśli na przykład komunikat został umieszczony przez odbierającego agenta MCA przy użyciu identyfikatora użytkownika agenta MCA, menedżer kolejek umieszcza raport COA przy użyciu identyfikatora użytkownika agenta MCA.

   Aplikacje generujące raporty powinny zwykle korzystać z tych samych uprawnień, które zostały użyte do wygenerowania odpowiedzi; powinno to być zwykle uprawnienie identyfikatora użytkownika w oryginalnym komunikacie.

   Jeśli raport musi być przemieszczany do zdalnego miejsca docelowego, nadawcy i odbiorcy mogą zdecydować, czy go zaakceptować, w taki sam sposób, jak w przypadku innych komunikatów.
4. Jeśli żądany jest komunikat raportu z danymi:

   - Komunikat raportu jest zawsze generowany z ilością danych żądanych przez nadawcę oryginalnego komunikatu. Jeśli komunikat raportu jest zbyt duży dla kolejki odpowiedzi, następuje przetwarzanie opisane wcześniej; komunikat raportu nigdy nie jest obcinany, aby zmieścił się w kolejce odpowiedzi.
   - Jeśli MDFMT oryginalnego komunikatu to FMXQH, dane uwzględnione w raporcie nie obejmują MQXQH. Dane raportu rozpoczynają się od pierwszego bajtu danych poza MQXQH w oryginalnym komunikacie. Dzieje się tak niezależnie od tego, czy kolejka jest kolejką transmisji.
5. Jeśli w kolejce odpowiedzi zostanie odebrany komunikat COA, COD lub raportu o utracie ważności, to jest pewne, że oryginalny komunikat został odebrany, dostarczony lub utracił ważność (w zależności od przypadku). Jeśli jednak co najmniej jeden z tych komunikatów raportu jest żądany i nie został odebrany, nie można przyjąć wartości odwrotnej, ponieważ mógł wystąpić jeden z następujących zdarzeń:

   a. Komunikat raportu jest wstrzymany, ponieważ odsyłacz jest wyłączony.

b. Komunikat raportu jest wstrzymany, ponieważ warunek blokowania istnieje w pośredniej kolejce transmisji lub w kolejce odpowiedzi (na przykład kolejka jest pełna lub zablokowana dla operacji umieszczania).

c. Komunikat raportu znajduje się w kolejce niedostarczonych komunikatów.

d. Gdy menedżer kolejek próbował wygenerować komunikat raportu, nie mógł umieścić go w odpowiedniej kolejce, a także nie mógł umieścić go w kolejce niedostarczonych komunikatów, dlatego nie można wygenerować komunikatu raportu.

e. Wystąpiło niepowodzenie menedżera kolejek między zgłoszonym działaniem (nadejście, dostarczenie lub utrata ważności) a wygenerowaniem odpowiedniego komunikatu raportu. (Nie dzieje się tak w przypadku komunikatów raportu COD, jeśli aplikacja pobiera oryginalny komunikat w jednostce pracy, ponieważ komunikat raportu COD jest generowany w tej samej jednostce pracy).

Komunikaty raportów o wyjątkach mogą być wstrzymane w ten sam sposób z powodów 1, 2 i 3 wcześniej. Jeśli jednak agent MCA nie może wygenerować komunikatu raportu o wyjątku (komunikat raportu nie może być umieszczony ani w kolejce odpowiedzi, ani w kolejce niedostarczonych komunikatów), oryginalny komunikat pozostaje w kolejce transmisji u nadawcy, a kanał jest zamykany. Ma to miejsce bez względu na to, czy komunikat raportu miał zostać wygenerowany na wysyłającym, czy odbierającym końcu kanału.

6. Jeśli pierwotny komunikat jest tymczasowo zablokowany (co powoduje wygenerowanie komunikatu raportu o wyjątku i umieszczenie oryginalnego komunikatu w kolejce niedostarczonych komunikatów), ale blokada zostanie skasowana, a aplikacja odczyta oryginalny komunikat z kolejki niedostarczonych komunikatów i umieści go ponownie w miejscu docelowym, mogą wystąpić następujące warunki:

   - Mimo że został wygenerowany komunikat raportu o wyjątku, oryginalny komunikat w końcu dotarł pomyślnie do miejsca docelowego.
   - W odniesieniu do pojedynczego oryginalnego komunikatu generowany jest więcej niż jeden komunikat raportu o wyjątku, ponieważ oryginalny komunikat może napotkać później inną blokadę.

Zgłaszanie komunikatów podczas umieszczania w temacie:

1. Raporty mogą być generowane podczas umieszczania komunikatu w temacie. Ten komunikat zostanie wysłany do wszystkich subskrybentów tematu, który może mieć wartość zero, jeden lub wiele. Należy to wziąć pod uwagę przy wyborze opcji raportu, ponieważ w wyniku tego może zostać wygenerowanych wiele komunikatów raportu.
2. Podczas umieszczania komunikatu w temacie może istnieć wiele kolejek docelowych, którym ma zostać nadana kopia komunikatu. Jeśli w niektórych z tych kolejek docelowych występuje problem, taki jak zapełnienie kolejki, pomyślne zakończenie operacji MQPUT zależy od ustawienia parametru NPMSGDLV lub PMSGDLV (w zależności od trwałości komunikatu). Jeśli ustawienie jest takie, że dostarczenie komunikatu do kolejki docelowej musi zakończyć się pomyślnie (na przykład jest to komunikat trwały do trwałego subskrybenta, a parametr PMSGDLV jest ustawiony na wartość ALL lub ALLDUR), to powodzenie jest definiowane jako jedno z następujących kryteriów:

   - Pomyślnie wstawiono do kolejki subskrybenta
   - Użycie RODLQ i pomyślne umieszczenie w kolejce niedostarczonych komunikatów, jeśli kolejka subskrybenta nie może pobrać komunikatu
   - Użycie RODISC, jeśli kolejka subskrybenta nie może pobrać komunikatu.

Raportowanie komunikatów dla segmentów komunikatów:

1. Komunikaty raportu mogą być żądane dla komunikatów, dla których dozwolona jest segmentacja (patrz opis flagi MFSEGA). Jeśli menedżer kolejek uzna, że konieczne jest podzielenie komunikatu na segmenty, może zostać wygenerowany komunikat raportu dla każdego z segmentów, który następnie napotka odpowiedni warunek. Dlatego aplikacje powinny być przygotowane do odbierania wielu komunikatów raportu dla każdego typu komunikatu raportu, którego dotyczy żądanie. Pole `MDGID` w komunikacie raportu może być używane do korelowania wielu raportów

z identyfikatorem grupy oryginalnego komunikatu oraz pole MDFB używane do identyfikowania typu każdego komunikatu raportu.

2. Jeśli komenda GMLOGO jest używana do pobierania komunikatów raportu dla segmentów, należy pamiętać, że kolejne wywołania MQGET mogą zwracać raporty różnych typów. Jeśli na przykład dla komunikatu segmentowanego przez menedżer kolejek zażądano zarówno raportów COA, jak i COD, wywołania MQGET dla komunikatów raportu mogą zwrócić komunikaty raportu COA i COD przeplatające się w nieprzewidywalny sposób. Można tego uniknąć, używając opcji GMCMPM (opcjonalnie z GMATM). Komenda GMCMPM powoduje, że menedżer kolejek ponownie składa komunikaty raportu tego samego typu. Na przykład pierwsze wywołanie MQGET może ponownie złożyć wszystkie komunikaty COA dotyczące oryginalnego komunikatu, a drugie wywołanie MQGET może ponownie złożyć wszystkie komunikaty COD. To, który komunikat jest składany jako pierwszy, zależy od typu komunikatu raportu, który występuje jako pierwszy w kolejce.
3. Aplikacje, które same umieszczają segmenty, mogą określać różne opcje raportu dla każdego segmentu. Należy jednak zwrócić uwagę na następujące kwestie:
   - Jeśli segmenty są pobierane przy użyciu opcji GMCMPM, tylko opcje raportu w pierwszym segmencie są uwzględniane przez menedżer kolejek.
   - Jeśli segmenty są pobierane jeden po drugim, a większość z nich ma jedną z opcji ROCOD*, ale co najmniej jeden segment nie, nie będzie możliwe użycie opcji GMCMPM do pobrania komunikatów raportu za pomocą pojedynczego wywołania MQGET lub użycie opcji GMASGA do wykrycia nadejścia wszystkich komunikatów raportu.
4. W sieci produktu MQ menedżery kolejek mogą mieć różne możliwości. Jeśli komunikat raportu dla segmentu jest generowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji, menedżer kolejek lub agent MCA domyślnie nie dołączą niezbędnych informacji o segmencie do komunikatu raportu, co może utrudnić zidentyfikowanie oryginalnego komunikatu, który spowodował wygenerowanie raportu. Można uniknąć tej trudności, wysyłając żądanie danych z komunikatem raportu, czyli podając odpowiednie opcje RO* D lub RO* F. Należy jednak pamiętać, że jeśli określono RO* D, mniej niż 100 bajtów danych komunikatu aplikacji może zostać zwróconych do aplikacji, która pobiera komunikat raportu, jeśli komunikat raportu został wygenerowany przez menedżer kolejek lub agent MCA, który nie obsługuje segmentacji.

Treść deskryptora komunikatu dla komunikatu raportu: gdy menedżer kolejek lub agent kanału komunikatów (MCA) generuje komunikat raportu, ustawia pola w deskryptorze komunikatu na następujące wartości, a następnie umieszcza komunikat w normalny sposób.

Tabela 708. Wartości używane w polach MQMD, gdy komunikat raportu jest generowany przez system

Pole w strukturze MQMD	Użyta wartość
MDSID	MDSIDV
MDVER	MDVER2
MDREP	BRAK
MDMT	MTRPRT
MDEXP	EIULIM
MDFB	W zależności od rodzaju raportu (FBCOA, FBCOD, FBEXP lub wartość RC*)
MDENC	Skopiowane z oryginalnego deskryptora komunikatu
MDCSI	Skopiowane z oryginalnego deskryptora komunikatu
MDFMT	Skopiowane z oryginalnego deskryptora komunikatu
MDPRI	Skopiowane z oryginalnego deskryptora komunikatu
MDPER	Skopiowane z oryginalnego deskryptora komunikatu
MDMID	Zgodnie z opcjami raportu w oryginalnym deskryptorze komunikatu
MDCID	Zgodnie z opcjami raportu w oryginalnym deskryptorze komunikatu

Tabela 708. Wartości używane w polach MQMD, gdy komunikat raportu jest generowany przez system (kontynuacja)

Pole w strukturze MQMD	Użyta wartość
MDBOC	0
MDRQ	Puste
MDRM	Nazwa menedżera kolejek.
MDUID	Zgodnie z opcją PMPASI
MDACC	Zgodnie z opcją PMPASI
MDAID	Zgodnie z opcją PMPASI
MDPAT	ATQM lub odpowiednio dla agenta kanału komunikatów
MDPAN	Pierwsze 28 bajtów nazwy menedżera kolejek lub nazwy agenta kanału komunikatów. W przypadku komunikatów raportu wygenerowanych przez most IMS to pole zawiera nazwę grupy XCF i nazwę elementu XCF systemu IMS , do którego odnosi się komunikat.
MDPD	Data wysłania komunikatu raportu
MDPT	Czas wysłania komunikatu raportu
MDAOD	Puste
MDGID	Skopiowane z oryginalnego deskryptora komunikatu
MDSEQ	Skopiowane z oryginalnego deskryptora komunikatu
MDOFF	Skopiowane z oryginalnego deskryptora komunikatu
MDMFL	Skopiowane z oryginalnego deskryptora komunikatu
MDOLN	Skopiowane z oryginalnego deskryptora komunikatu, jeśli nie jest to OLUNDF, i ustawione na długość oryginalnych danych komunikatu, w przeciwnym razie

Zaleca się, aby aplikacja generująca raport ustawiła podobne wartości, z wyjątkiem następujących:

- Pole MDRM można ustawić na wartość pustą (menedżer kolejek zmieni tę wartość na nazwę lokalnego menedżera kolejek po umieszczeniu komunikatu).
- Pola kontekstu powinny być ustawione przy użyciu opcji, która zostałaby użyta dla odpowiedzi, zwykle PMPASI.

Analizowanie pola raportu: Pole MDREP zawiera podpola. Z tego powodu aplikacje, które muszą sprawdzić, czy nadawca komunikatu, którego dotyczy żądanie, powinien użyć jednej z technik opisanych w sekcji "Analizowanie pola raportu w systemie IBM i" na stronie 1480.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Wartością początkową tego pola jest RONONE.

MDRM (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek odpowiedzi.

Jest to nazwa menedżera kolejek, do którego ma zostać wysłany komunikat odpowiedzi lub raport. MDRQ to nazwa lokalna kolejki, która jest zdefiniowana w tym menedżerze kolejek.

Jeśli pole MDRM jest puste, menedżer kolejek lokalnych wyszukuje nazwę **MDRQ** w swoich definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość **MDRM** w przesłanym komunikacie jest zastępowana wartością atrybutu **RemoteQMgrName** z definicji kolejki zdalnej i ta wartość zostanie zwrócona w deskryptorze komunikatu, gdy aplikacja odbierająca wyśle wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, MDRM przesyłany z komunikatem jest nazwą lokalnego menedżera kolejek.

Jeśli nazwa jest określona, może zawierać końcowe odstępy; pierwszy znak o kodzie zero i następujące po nim znaki są traktowane jako odstępy. W przeciwnym razie nie jest wykonywane żadne sprawdzenie, czy nazwa jest zgodna z regułami nazewnictwa menedżerów kolejek lub czy ta nazwa jest rozpoznawana przez nadawczy menedżer kolejek. Ta sytuacja ma również zastosowanie w przypadku nazwy przesłanej, jeśli nazwa **MDRM** została zastąpiona w przesłanym komunikacie.

Jeśli kolejka odpowiedzi nie jest wymagana, zaleca się (chociaż nie jest to zaznaczone), aby pole MDRM było puste; pole nie powinno pozostać niezainicjowane.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnianą spacjami do długości pola.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest określona przez LNQMN. Wartością początkową tego pola jest 48 znaków odstępu.

MDRQ (48-bajtowy łańcuch znaków)

Nazwa kolejki odpowiedzi.

Jest to nazwa kolejki komunikatów, do której aplikacja, która wysłała żądanie pobrania komunikatu, powinna wysłać komunikaty MTRPLY i MTRPRT. Nazwa jest nazwą lokalną kolejki, która jest zdefiniowana w menedżerze kolejek identyfikowanym przez MDRM. Ta kolejka nie powinna być kolejką modelową, chociaż nadawczy menedżer kolejek nie sprawdza tego podczas umieszczania komunikatu.

W przypadku wywołań MQPUT i MQPUT1 to pole nie może być puste, jeśli pole MDMT ma wartość MTRQST lub jeśli w polu MDREP są żądane komunikaty raportu. Jednak określona (lub podstawiona) wartość jest przekazywana do aplikacji, która wysyła żądanie pobrania dla komunikatu, niezależnie od typu komunikatu.

Jeśli pole MDRM jest puste, menedżer kolejek lokalnych wyszukuje nazwę MDRQ we własnych definicjach kolejek. Jeśli istnieje lokalna definicja kolejki zdalnej o tej nazwie, wartość MDRQ w przesłanym komunikacie jest zastępowana wartością atrybutu **RemoteQName** z definicji kolejki zdalnej i ta wartość zostanie zwrócona w deskryptorze komunikatu, gdy aplikacja odbierająca wyśle wywołanie MQGET dla komunikatu. Jeśli lokalna definicja kolejki zdalnej nie istnieje, wartość MDRQ pozostaje niezmieniona.

Jeśli nazwa jest określona, może zawierać końcowe odstępy; pierwszy znak o kodzie zero i następujące po nim znaki są traktowane jako odstępy. W przeciwnym razie jednak nie jest wykonywane żadne sprawdzenie, czy nazwa spełnia reguły nazewnictwa dla kolejek. Jest to również prawdą dla przesyłanej nazwy, jeśli nazwa MDRQ jest zastępowana w przesyłanym komunikacie. Jedynym sprawdzonym sposobem jest określenie nazwy, jeśli okoliczności tego wymagają.

Jeśli kolejka odpowiedzi nie jest wymagana, zaleca się (chociaż nie jest to zaznaczone), aby pole MDRQ było puste; pole nie powinno pozostać niezainicjowane.

W przypadku wywołania MQGET menedżer kolejek zawsze zwraca nazwę dopełnianą spacjami do długości pola.

Jeśli komunikat, który wymaga komunikatu raportu, nie może zostać dostarczony, a komunikat raportu nie może zostać dostarczony do określonej kolejki, zarówno oryginalny komunikat, jak i komunikat raportu są umieszczane w kolejce niedostarczonych komunikatów. Patrz atrybut **DeadLetterQName** opisany w sekcji “Atrybuty menedżera kolejek w systemie IBM i” na stronie 1443.

Jest to pole wyjściowe dla wywołania MQGET i pole wejściowe dla wywołań MQPUT i MQPUT1 . Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

MDSEQ (10-cyfrowa liczba całkowita ze znakiem)

Numer kolejny komunikatu logicznego w grupie.

Numery kolejne zaczynają się od 1 i zwiększają się o 1 dla każdego nowego komunikatu logicznego w grupie, maksymalnie do 999 999 999. Komunikat fizyczny, który nie znajduje się w grupie, ma numer kolejny 1.

To pole nie musi być ustawiane przez aplikację w wywołaniu MQPUT lub MQGET, jeśli:

- W wywołaniu MQPUT określono PMLOGO.
- W wywołaniu MQGET nie określono parametru MOSEQN.

Są to zalecane sposoby używania tych wywołań dla komunikatów, które nie są komunikatami raportów. Jeśli jednak aplikacja wymaga większej kontroli lub wywołanie ma wartość MQPUT1, aplikacja musi upewnić się, że parametr MDSEQ ma odpowiednią wartość.

Na wejściu wywołań MQPUT i MQPUT1 menedżer kolejek używa wartości określonej w Tabeli 1. W danych wyjściowych wywołań MQPUT i MQPUT1 menedżer kolejek ustawia w tym polu wartość, która została wysłana razem z komunikatem.

Na wejściu wywołania MQGET menedżer kolejek używa wartości określonej w Tabeli 1. W danych wyjściowych wywołania MQGET menedżer kolejek ustawia w tym polu wartość pobranego komunikatu.

Wartością początkową tego pola jest jeden. To pole jest ignorowane, jeśli wartość MDVER jest mniejsza niż MDVER2.

MDSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

MDSIDV
Identyfikator struktury deskryptora komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MDSIDV.

MDUID (12-bajtowy łańcuch znaków)

Identyfikator użytkownika.

Jest to część *kontekstu tożsamości* komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

MDUID określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu.

Po odebraniu komunikatu można użyć pola MDUID w polu ODAU parametru **OBJDSC** kolejnego wywołania MQOPEN lub MQPUT1 , aby sprawdzenie autoryzacji było wykonywane dla użytkownika MDUID , a nie dla aplikacji wykonującej operację otwierania.

Gdy menedżer kolejek generuje te informacje dla wywołania MQPUT lub MQPUT1 , używa on identyfikatora użytkownika określonego na podstawie środowiska.

Jeśli identyfikator użytkownika jest określany na podstawie środowiska:

- z/OS W systemie z/OSmenedżer kolejek używa:
  - W przypadku zadania wsadowego identyfikator użytkownika z karty JES JOB lub uruchomionego zadania
  - W przypadku TSO-identyfikator logowania użytkownika
  - W przypadku systemu CICSjest to identyfikator użytkownika powiązany z zadaniem.
  - W przypadku systemu IMSidentyfikator użytkownika zależy od typu aplikacji:
    - Przez:
      - Regiony BMP bez komunikatów
      - Regiony IFP niebędące wiadomością
      - Regiony BMP komunikatu i IFP komunikatu, które nie wywołały pomyślnego wywołania GU

Menedżer kolejek używa identyfikatora użytkownika z karty JES JOB regionu lub identyfikatora użytkownika TSO. Jeśli są one puste lub mają wartość null, używana jest nazwa bloku specyfikacji programu (PSB).

- Przez:
  - Regiony BMP komunikatu i IFP komunikatu, które pomyślnie wywołały GU
  - Regiony MPP

  menedżer kolejek używa jednej z następujących wartości:
  - Identyfikator zalogowanego użytkownika powiązany z komunikatem
  - Nazwa terminalu logicznego (LTERM)
  - Identyfikator użytkownika z karty JES JOB regionu
  - Identyfikator użytkownika TSO
  - Nazwa PSB

- IBM i W systemie IBM imenedżer kolejek używa nazwy profilu użytkownika powiązanego z zadaniem aplikacji.
- Linux AIX W systemie AIX and Linuxmenedżer kolejek używa:
  - Nazwa logowania aplikacji
  - Efektywny identyfikator użytkownika procesu, jeśli nie jest dostępne logowanie
  - Identyfikator użytkownika powiązany z transakcją, jeśli aplikacja jest transakcją CICS
- W systemie VSE/ESAjest to pole zastrzeżone.
- Windows W systemie Windowsmenedżer kolejek używa pierwszych 12 znaków nazwy zalogowanego użytkownika.

W przypadku wywołań MQPUT i MQPUT1 jest to pole wejściowe/wyjściowe, jeśli w parametrze **PMO** określono PMSETI lub PMSETA. Wszelkie informacje następujące po znaku o kodzie zero w polu są odrzucane. Znak o kodzie zero i wszystkie następujące po nim znaki są przekształcane przez menedżera kolejek w odstępy. Jeśli nie określono PMSETI lub PMSETA, to pole jest ignorowane na wejściu i jest polem tylko wyjściowym.

Po pomyślnym zakończeniu wywołania MQPUT lub MQPUT1 to pole zawiera wartość `MDUID` , która została przesłana wraz z komunikatem, jeśli został on umieszczony w kolejce. Będzie to wartość parametru `MDUID` , która jest przechowywana razem z komunikatem, jeśli zostanie zachowana (więcej informacji na temat zachowanych publikacji zawiera opis PMRET), ale nie jest używana jako parametr `MDUID` , gdy komunikat jest wysyłany jako publikacja do subskrybentów, ponieważ we wszystkich wysłanych do nich publikacjach podano wartość przesłaniającą parametr `MDUID` . Jeśli komunikat nie ma kontekstu, pole jest całkowicie puste.

Jest to pole wyjściowe wywołania MQGET. Długość tego pola jest określona przez LNUID. Wartością początkową tego pola jest 12 pustych znaków.

MDVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

MDVER1
Struktura deskryptora komunikatu Version-1 .

MDVER2
Struktura deskryptora komunikatu Version-2 .

Uwaga: Jeśli używana jest struktura MQMD w wersji version-2 , menedżer kolejek wykonuje dodatkowe sprawdzenia dotyczące wszystkich struktur nagłówków MQ , które mogą być obecne

na początku danych komunikatu aplikacji. Szczegółowe informacje można znaleźć w uwagach dotyczących składni wywołania MQPUT.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MDVERC
Bieżąca wersja struktury deskryptora komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MDVER1.

Wartości początkowe

Tabela 709. Pola w strukturze MQMD

Nazwa pola	Nazwa stałej	Wartość stałej
MDSID	MDSIDV	'MD¬¬'
MDVER	MDVER1	1
MDREP	BRAK	0
MDMT	MTGRM	8
MDEXP	EIULIM	-1
MDFB	FBNONE	0
MDENC	ENNAT,	Zależy od środowiska
MDCSI	CSQM	0
MDFMT	BRAK FMNONE	Puste
MDPRI	PRQDEF,	-1
MDPER	PEQDEF,	2
MDMID	MINONE	Wartości null
MDCID	KINON	Wartości null
MDBOC	Brak	0
MDRQ	Brak	Puste
MDRM	Brak	Puste
MDUID	Brak	Puste
MDACC	ACNONE (brak)	Wartości null
MDAID	Brak	Puste
MDPAT	ATNCON	0
MDPAN	Brak	Puste
MDPD	Brak	Puste
MDPT	Brak	Puste
MDAOD	Brak	Puste
MDGID	GINON	Wartości null
MDSEQ	Brak	1
MDOFF	Brak	0

Tabela 709. Pola w strukturze MQMD (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
MDMFL	BRAK MFNONE	0
MDOLN	OLUNDF	-1
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQMD Structure
D*
D* Structure identifier
D  MDSID                  1      4    INZ('MD  ')
D* Structure version number
D  MDVER                  5      8I 0 INZ(1)
D* Options for report messages
D  MDREP                  9     12I 0 INZ(0)
D* Message type
D  MDMT                  13     16I 0 INZ(8)
D* Message lifetime
D  MDEXP                 17     20I 0 INZ(-1)
D* Feedback or reason code
D  MDFB                  21     24I 0 INZ(0)
D* Numeric encoding of message data
D  MDENC                 25     28I 0 INZ(273)
D* Character set identifier of messagedata
D  MDCSI                 29     32I 0 INZ(0)
D* Format name of message data
D  MDFMT                 33     40    INZ('        ')
D* Message priority
D  MDPRI                 41     44I 0 INZ(-1)
D* Message persistence
D  MDPER                 45     48I 0 INZ(2)
D* Message identifier
D  MDMID                 49     72    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000')
D* Correlation identifier
D  MDCID                 73     96    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000')
D* Backout counter
D  MDBOC                 97    100I 0 INZ(0)
D* Name of reply queue
D  MDRQ                 101    148    INZ
D* Name of reply queue manager
D  MDRM                 149    196    INZ
D* User identifier
D  MDUID                197    208    INZ
D* Accounting token
D  MDACC                209    240    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000000000000-
D                                     000000')
D* Application data relating toidentity
D  MDAID                241    272    INZ
D* Type of application that put themessage
D  MDPAT                273    276I 0 INZ(0)
D* Name of application that put themessage
D  MDPAN                277    304    INZ
D* Date when message was put
D  MDPD                 305    312    INZ
D* Time when message was put
D  MDPT                 313    320    INZ
D* Application data relating toorigin
D  MDAOD                321    324    INZ
D* Group identifier
D  MDGID                325    348    INZ(X'00000000000000-
D                                     000000000000000000000-
```

```
D                                       000000000000')
D* Sequence number of logical messagewithin group
D  MDSEQ                349    352I 0 INZ(1)
D* Offset of data in physical messagefrom start of logical message
D  MDOFF                353    356I 0 INZ(0)
D* Message flags
D  MDMFL                357    360I 0 INZ(0)
D* Length of original message
D  MDOLN                361    364I 0 INZ(-1)
```

IBM i MQMDE (Rozszerzenie deskryptora komunikatu) w systemie IBM i

Przegląd

Cel: Struktura MQMDE opisuje dane, które czasami występują przed danymi komunikatu aplikacji. Struktura zawiera te pola MQMD, które istnieją w version-2 MQMD, ale nie w version-1 MQMD.

Nazwa formatu: FMMDE.

Zestaw znaków i kodowanie: dane w MQMDE muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek systemu **CodedCharSetId** oraz kodowanie lokalnego menedżera kolejek określone przez ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQMDE muszą być ustawione w polach *MDCSI* i *MDENC* w następujących polach:

- MQMD (jeśli struktura MQMDE jest na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQMDE (wszystkie inne przypadki).

Jeśli produkt MQMDE nie znajduje się w zestawie znaków i kodowaniu menedżera kolejek, produkt MQMDE jest akceptowany, ale nie jest honorowany, co oznacza, że produkt MQMDE jest traktowany jako dane komunikatu.

Użycie: Normalne aplikacje powinny używać deskryptora MQMD version-2 , w którym to przypadku nie napotkają struktury MQMDE. Jednak w niektórych sytuacjach wyspecjalizowane aplikacje i aplikacje, które nadal używają deskryptora MQMD version-1 , mogą napotkać interfejs MQMDE. Struktura MQMDE może wystąpić w następujących okolicznościach:

- Określone w wywołaniach MQPUT i MQPUT1
- Zwracane przez wywołanie MQGET
- W komunikatach w kolejkach transmisji

MQMDE określone w wywołaniach MQPUT i MQPUT1

W wywołaniach MQPUT i MQPUT1 , jeśli aplikacja udostępnia deskryptor MQMD version-1 , aplikacja może opcjonalnie poprzedzić dane komunikatu przedrostkiem MQMDE, ustawiając pole *MDFMT* w deskryptorze MQMD na wartość FMMDE w celu wskazania, że istnieje MQMDE. Jeśli aplikacja nie udostępnia środowiska MQMDE, menedżer kolejek przyjmuje wartości domyślne dla pól w środowisku MQMDE. Wartości domyślne używane przez menedżer kolejek są takie same jak wartości początkowe struktury-patrz sekcja Tabela 711 na stronie 1191.

Jeśli aplikacja udostępnia version-2 MQMD *i* poprzedza dane komunikatu aplikacji przedrostkiem MQMDE, struktury są przetwarzane zgodnie z opisem w sekcji Tabela 710 na stronie 1188.

Tabela 710. Działanie menedżera kolejek po podaniu MQMDE w MQPUT lub MQPUT1			
Wersja MQMD	**Wartości pól version-2**	**Wartości odpowiednich pól w MQMDE**	**Działanie wykonywane przez menedżer kolejek**
1	-	Ważne	MQMDE jest honorowane
2	Domyślny	Ważne	MQMDE jest honorowane
2	Niedomyślna	Ważne	MQMDE jest traktowane jako dane komunikatu
1 lub 2	Dowolna	Niepoprawne	Wywołanie nie powiodło się z odpowiednim kodem przyczyny
1 lub 2	Dowolna	MQMDE jest w niepoprawnym zestawie znaków lub kodowaniu albo jest nieobsługiwaną wersją	MQMDE jest traktowane jako dane komunikatu

Jest jeden szczególny przypadek. Jeśli aplikacja używa deskryptora MQMD version-2 w celu umieszczenia komunikatu, który jest segmentem (flaga MFSEG lub MFLSEG jest ustawiona), a nazwa formatu w deskryptorze MQMD to FMDLH, menedżer kolejek generuje strukturę MQMDE i wstawia ją *między* strukturą MQDLH a danymi, które następują po niej. W strukturze MQMD, w której menedżer kolejek zachowuje komunikat, pola version-2 są ustawione na wartości domyślne.

Niektóre z pól, które istnieją w programie MQMD version-2 , ale nie w programie MQMD version-1 , są polami wejściowymi/wyjściowymi MQPUT i MQPUT1. Jednak menedżer kolejek nie zwraca żadnych wartości w odpowiednich polach w MQMDE na wyjściu z wywołań MQPUT i MQPUT1 . Jeśli aplikacja wymaga tych wartości wyjściowych, musi użyć deskryptora MQMD version-2 .

MQMDE zwrócone przez wywołanie MQGET

W wywołaniu MQGET, jeśli aplikacja udostępnia deskryptor MQMD version-1 , menedżer kolejek dodaje przedrostek do komunikatu zwróconego z MQMDE, ale tylko wtedy, gdy co najmniej jedno z pól w MQMDE ma wartość inną niż domyślna. Menedżer kolejek ustawia pole *MDFMT* w strukturze MQMD na wartość FMMDE, aby wskazać, że istnieje struktura MQMDE.

Jeśli aplikacja udostępnia środowisko MQMDE na początku parametru **BUFFER** , środowisko MQMDE jest ignorowane. Po powrocie z wywołania MQGET jest on zastępowany przez MQMDE dla komunikatu (jeśli jest potrzebny) lub nadpisywany przez dane komunikatu aplikacji (jeśli nie jest potrzebny MQMDE).

Jeśli MQMDE jest zwracane przez wywołanie MQGET, dane w MQMDE zwykle znajdują się w zestawie znaków i kodowaniu menedżera kolejek. Jednak MQMDE może być w innym zestawie znaków i kodowaniu, jeśli:

- Środowisko MQMDE było traktowane jako dane w wywołaniu MQPUT lub MQPUT1 (informacje na temat okoliczności, które mogą spowodować tę sytuację, zawiera sekcja Tabela 710 na stronie 1188).
- Komunikat został odebrany od menedżera kolejek zdalnych połączonego z połączeniem TCP, a agent kanału komunikatów odbierających (MCA) nie został poprawnie skonfigurowany (więcej informacji na ten temat zawiera sekcja Zabezpieczenia obiektów produktu IBM MQ for IBM i).

MQMDE w komunikatach w kolejkach transmisji

Komunikaty w kolejkach transmisji są poprzedzone strukturą MQXQH, która zawiera w sobie strukturę MQMD version-1 . Środowisko MQMDE może być również obecne, umieszczone między strukturą MQXQH i danymi komunikatu aplikacji, ale zwykle będzie obecne tylko wtedy, gdy co najmniej jedno pole w środowisku MQMDE ma wartość inną niż domyślna.

Inne struktury nagłówka IBM MQ mogą również występować między strukturą MQXQH a danymi komunikatu aplikacji. Jeśli na przykład istnieje nagłówek niedostarczonego komunikatu MQDLH, a komunikat nie jest segmentem, kolejność jest następująca:

- MQXQH (zawierający version-1 MQMD)
- MQMDE,
- MQDLH
- Dane komunikatu aplikacji

Pola

Struktura MQMDE zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

MECSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych następujący po MQMDE.

Określa identyfikator zestawu znaków danych, które są zgodne ze strukturą MQMDE. Nie ma on zastosowania do danych znakowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Można użyć następującej wartości specjalnej:

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola *MDPAT* w MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

MEENC (10-cyfrowa liczba całkowita ze znakiem)
MEENC (10-cyfrowa liczba całkowita ze znakiem)

Określa kodowanie liczbowe danych, które są zgodne ze strukturą MQMDE. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy pole jest poprawne. Więcej informacji na temat kodowania danych zawiera opis pola *MDENC* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141 .

Wartością początkową tego pola jest ENNAT.

MEFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi ogólne.

Można podać następującą opcję:

MEFNON
Brak flag.

Wartością początkową tego pola jest MEFNON.

MEFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych następujących po MQMDE.

Określa nazwę formatu danych, które są zgodne ze strukturą MQMDE.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Menedżer kolejek nie sprawdza, czy to pole jest poprawne. Więcej informacji na temat nazw formatów zawiera opis pola *MDFMT* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141 .

Wartością początkową tego pola jest FMNONE.

MEGID (24-bajtowy łańcuch bitowy)

Identyfikator grupy.

Patrz opis pola *MDGID* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.
Wartością początkową tego pola jest GINONE.

MELEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQMDE.

Zdefiniowana jest następująca wartość:

MELEN2
Długość struktury rozszerzenia deskryptora komunikatu version-2 .

Wartością początkową tego pola jest MELEN2.

MEMFL (10-cyfrowa liczba całkowita ze znakiem)

Flagi komunikatów.

Patrz opis pola *MDMFL* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.
Wartością początkową tego pola jest MFNONE.

MEOFF (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego.

Patrz opis pola *MDOFF* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.
Wartością początkową tego pola jest 0.

MEOLN (10-cyfrowa liczba całkowita ze znakiem)

Długość oryginalnego komunikatu.

Patrz opis pola *MDOLN* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.
Wartością początkową tego pola jest OLUNDF.

MESEQ (10-cyfrowa liczba całkowita ze znakiem)

Numer kolejny komunikatu logicznego w grupie.

Patrz opis pola *MDSEQ* w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.
Wartością początkową tego pola jest 1.

MESID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

MESIDV
Identyfikator struktury rozszerzenia deskryptora komunikatu.

Wartością początkową tego pola jest MESIDV.

MEVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

MEVER2
Struktura rozszerzenia deskryptora komunikatu Version-2 .

Następująca stała określa numer wersji bieżącej:

MEVERC (średnia)
Bieżąca wersja struktury rozszerzenia deskryptora komunikatu.

Wartością początkową tego pola jest MEVER2.

Wartości początkowe

Tabela 711. Pola w MQMDE		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
MESID	MESIDV	'MDE¬'
MEVER	MEVER2	2
MELEN	MELEN2	72
MEENC	ENNAT,	Zależy od środowiska
MECSI	CSUNDF	0
MEFMT	BRAK FMNONE	Puste
MEFLG	MEFNON	0
MEGID	GINON	Wartości null
MESEQ	Brak	1
MEOFF	Brak	0
MEMFL	BRAK MFNONE	0
MEOLN	OLUNDF	-1
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQMDE Structure
D*
D* Structure identifier
D  MESID                  1      4    INZ('MDE ')
D* Structure version number
D  MEVER                  5      8I 0 INZ(2)
D* Length of MQMDE structure
D  MELEN                  9     12I 0 INZ(72)
D* Numeric encoding of data that followsMQMDE
D  MEENC                 13     16I 0 INZ(273)
D* Character-set identifier of data thatfollows MQMDE
D  MECSI                 17     20I 0 INZ(0)
D* Format name of data that followsMQMDE
D  MEFMT                 21     28    INZ('        ')
D* General flags
D  MEFLG                 29     32I 0 INZ(0)
D* Group identifier
D  MEGID                 33     56    INZ(X'00000000000000-
D                                     000000000000000000000-
D                                     000000000000')
D* Sequence number of logical messagewithin group
D  MESEQ                 57     60I 0 INZ(1)
D* Offset of data in physical messagefrom start of logical message
D  MEOFF                 61     64I 0 INZ(0)
D* Message flags
D  MEMFL                 65     68I 0 INZ(0)
D* Length of original message
D  MEOLN                 69     72I 0 INZ(-1)
```

IBM i MQMHBO (Message handle to buffer options) w systemie IBM i

Struktura definiująca uchwyt komunikatu dla opcji buforu

Przegląd

Cel: Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia buforów z uchwytów komunikatów. Struktura jest parametrem wejściowym wywołania MQMHBUF.

Zestaw znaków i kodowanie: Dane w MQMHBO muszą być w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1192
- "Wartości początkowe" na stronie 1193
- "Deklaracja RPG" na stronie 1193

Pola

Struktura MQMHBO zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

MBOPT (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt komunikatu do struktury opcji buforu-pole MBOPT.

Te opcje sterują działaniem MQMHBUF.

Należy podać następującą opcję:

MBPRRF
Podczas przekształcania właściwości z uchwytu komunikatu w bufor należy przekształcić je w format MQRFH2 .

Opcjonalnie można również określić następującą opcję. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

MDLPR
Właściwości dodawane do buforu są usuwane z uchwytu komunikatu. Jeśli wywołanie nie powiedzie się, nie zostaną usunięte żadne właściwości.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MBPRRF.

MBSID (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt komunikatu do struktury opcji buforu-pole MBSID.

Jest to identyfikator struktury. Wartość musi być następująca:

MBSIDV
Identyfikator struktury opcji buforu dla uchwytu komunikatu.

Jest to zawsze pole wejściowe. Początkowa wartość w tym polu isMBSIDV.

MBVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

MBVER1
Numer wersji struktury opcji przesyłania komunikatu do buforu.

Następująca stała określa numer wersji bieżącej:

MBVERC
Bieżąca wersja struktury opcji przesyłania komunikatów do buforu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest MBVER1.

Wartości początkowe

Tabela 712. Pola w MQMHBO		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
MVSID	MBSIDV	'MHBO'
MBVER	MBVER1	1
MBOPT	MBPRRF	

Uwagi:

1. Wartość Null lub odstępy oznaczają znak odstępu.

Deklaracja RPG

```
D* MQMHBO Structure
D*
D*
D* Structure identifier
D  MBSID                  1      4    INZ('MHBO')
D*
D* Structure version number
D  MBVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQMHBUF
D  MBOPT                  9     12I 0 INZ(1)
```

IBM i MQOD (deskryptor obiektu) w systemie IBM i

Struktura MQOD służy do określania obiektu według nazwy.

Przegląd

Przeznaczenie: Poprawne są następujące typy obiektów:

- Kolejka lub lista dystrybucyjna
- Lista nazw
- Definicja procesu
- Menedżer kolejek
- Temat

Struktura jest parametrem wejścia/wyjścia w wywołaniach MQOPEN i MQPUT1 .

Wersja: Bieżąca wersja produktu MQOD to ODVER4. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Udostępniony plik COPY zawiera najnowszą wersję programu MQOD, która jest obsługiwana przez środowisko, ale z wartością początkową pola *ODVER* ustawioną na ODVER1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *ODVER* numer wersji wymaganej wersji.

Aby otworzyć listę dystrybucyjną, *ODVER* musi mieć wartość ODVER2 lub większą.

Zestaw znaków i kodowanie: Dane w programie MQOD muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek produktu **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez translację ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

- “Pola” na stronie 1194
- “Wartości początkowe” na stronie 1201
- “Deklaracja RPG” na stronie 1202

Pola

Struktura MQOD zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

ODASI (40-bajtowy łańcuch bitowy)

Alternatywny identyfikator zabezpieczeń.

Jest to identyfikator bezpieczeństwa, który jest przekazywany z *ODAU* do usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji. Parametr *ODASI* jest używany tylko wtedy, gdy:

- OOALTU jest określone w wywołaniu MQOPEN lub
- Parametr MALTU jest określony w wywołaniu MQPUT1 ,

i pole *ODAU* nie jest całkowicie puste aż do pierwszego znaku o kodzie zero lub końca pola.

Pole *ODASI* ma następującą strukturę:

- Pierwszy bajt jest binarną liczbą całkowitą zawierającą długość istotnych danych, które następują po nim; wartość ta wyklucza sam bajt długości. Jeśli nie ma identyfikatora zabezpieczeń, długość wynosi zero.
- Drugi bajt wskazuje typ identyfikatora zabezpieczeń, który jest obecny; możliwe są następujące wartości:

  SITWNT
  Identyfikator zabezpieczeń Windows .

  SITNON
  Brak identyfikatora zabezpieczeń.
- Trzeci i kolejny bajt do długości określonej przez pierwszy bajt zawiera sam identyfikator bezpieczeństwa.
- Pozostałe bajty w polu są ustawione na zero binarne.

Można użyć następującej wartości specjalnej:

SINON
Nie określono identyfikatora zabezpieczeń.

Wartością długości pola jest zero binarne.

Jest to pole wejściowe. Długość tego pola jest określona przez LNSCID. Wartością początkową tego pola jest SINONE. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER3.

ODAU (12-bajtowy łańcuch znaków)

Alternatywny identyfikator użytkownika.

Jeśli określono wartość OOALTU dla wywołania MQOPEN lub PMALTU dla wywołania MQPUT1 , to pole zawiera alternatywny identyfikator użytkownika, który ma być używany do sprawdzania autoryzacji dla otwarcia zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona. Niektóre kontrole są jednak nadal przeprowadzane przy użyciu bieżącego identyfikatora użytkownika (na przykład sprawdzenia kontekstu).

Jeśli wartości OOALTU i PMALTU nie są określone i pole to jest całkowicie puste aż do pierwszego znaku o kodzie zero lub końca pola, operacja otwierania może zakończyć się powodzeniem tylko wtedy, gdy do otwarcia tego obiektu z podanymi opcjami nie jest wymagana autoryzacja użytkownika.

Jeśli nie określono wartości OOALTU ani PMALTU, to pole jest ignorowane.

Jest to pole wejściowe. Długość tego pola jest określona przez LNUID. Wartością początkową tego pola jest 12 pustych znaków.

ODDN (48-bajtowy łańcuch znaków)

Nazwa kolejki dynamicznej.

Jest to nazwa kolejki dynamicznej, która ma zostać utworzona przez wywołanie MQOPEN. Ma to znaczenie tylko wtedy, gdy parametr *ODON* określa nazwę kolejki modelowej; we wszystkich pozostałych przypadkach parametr *ODDN* jest ignorowany.

Znaki, które są poprawne w nazwie, są takie same jak w przypadku nazwy *ODON*, z tą różnicą, że gwiazdka jest również poprawna. Nazwa, która jest pusta (lub taka, w której przed pierwszym znakiem o kodzie zero wyświetlane są tylko odstępy), jest niepoprawna, jeśli *ODON* jest nazwą kolejki modelowej.

Jeśli ostatnim niepustym znakiem w nazwie jest gwiazdka (*), menedżer kolejek zastępuje gwiazdkę łańcuchem znaków, który gwarantuje, że nazwa wygenerowana dla kolejki jest unikalna w lokalnym menedżerze kolejek. Aby zapewnić wystarczającą liczbę znaków, gwiazdka jest poprawna tylko na pozycjach od 1 do 33. Po znaku gwiazdki nie mogą występować żadne znaki inne niż spacje ani znaki o kodzie zero.

Gwiazdka może występować na pierwszej pozycji znaku, w którym to przypadku nazwa składa się wyłącznie ze znaków wygenerowanych przez menedżer kolejek.

Jest to pole wejściowe. Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest `'AMQ.*'`, dopełniona odstępami.

ODIDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba kolejek, których otwarcie nie powiodło się.

Jest to liczba kolejek na liście dystrybucyjnej, których otwarcie nie powiodło się. Jeśli istnieje, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Uwaga: Jeśli istnieje, to pole jest ustawiane tylko wtedy, gdy parametr **CMPCOD** w wywołaniu MQOPEN lub MQPUT1 ma wartość CCOK lub CCWARN. Nie jest ustawiane, jeśli parametr **CMPCOD** ma wartość CCFAIL.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODKDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba pomyślnie otwartych kolejek lokalnych.

Jest to liczba kolejek na liście dystrybucyjnej, które są tłumaczone na kolejki lokalne i które zostały pomyślnie otwarte. Liczba ta nie obejmuje kolejek, które są tłumaczone na kolejki zdalne (nawet jeśli początkowo do przechowywania komunikatu używana jest lokalna kolejka transmisji). Jeśli istnieje, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODMN (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek obiektu.

Jest to nazwa menedżera kolejek, w którym zdefiniowano obiekt *ODON* . Znaki poprawne w nazwie są takie same jak w przypadku nazwy *ODON* (patrz wcześniej). Nazwa, która jest całkowicie pusta aż do pierwszego znaku o kodzie zero lub końca pola, oznacza menedżera kolejek, z którym połączona jest aplikacja (menedżer kolejek lokalnych).

Do wskazanych typów obiektów mają zastosowanie następujące punkty:

- Jeśli parametr *ODOT* ma wartość OTTOP, OTNLST, OTPRO lub OTQM, parametr *ODMN* musi być pusty lub musi mieć nazwę menedżera kolejek lokalnych.
- Jeśli *ODON* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej i zwraca w polu *ODMN* nazwę menedżera kolejek, w którym została utworzona kolejka. Jest to nazwa lokalnego menedżera kolejek. Kolejkę modelową można określić tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1 .

- Jeśli parametr *ODON* jest nazwą kolejki klastra, a parametr *ODMN* jest pusty, rzeczywiste miejsce docelowe komunikatów wysłanych przy użyciu uchwytu kolejki zwróconego przez wywołanie MQOPEN jest wybierane przez menedżer kolejek (lub wyjście obciążenia klastra, jeśli zostało zainstalowane) w następujący sposób:
  - Jeśli określono parametr OOBNDO, menedżer kolejek wybiera instancję kolejki klastra podczas przetwarzania wywołania MQOPEN, a wszystkie komunikaty umieszczone za pomocą tego uchwytu kolejki są wysyłane do tej instancji.
  - Jeśli określono OOBNDN, menedżer kolejek może wybrać inną instancję kolejki docelowej (rezydującą w innym menedżerze kolejek w klastrze) dla każdego kolejnego wywołania MQPUT używającego tego uchwytu kolejki.

  Jeśli aplikacja musi wysłać komunikat do *konkretnej* instancji kolejki klastra (czyli instancji kolejki rezydującej w określonym menedżerze kolejek w klastrze), w polu *ODMN* należy podać nazwę tego menedżera kolejek. Powoduje to, że lokalny menedżer kolejek wysyła komunikat do określonego docelowego menedżera kolejek.
- Jeśli otwierany obiekt jest listą dystrybucyjną (*ODREC* jest większe od zera), *ODMN* musi być pusty lub musi być łańcuchem pustym. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2153.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ODON* jest nazwą kolejki modelowej, a we wszystkich innych przypadkach jest to pole tylko wejściowe. Długość tego pola jest określona przez LNQMN. Wartością początkową tego pola jest 48 znaków odstępu.

ODON (48-bajtowy łańcuch znaków)

Nazwa obiektu.

Jest to nazwa lokalna obiektu zdefiniowana w menedżerze kolejek identyfikowanym przez *ODMN*. Nazwa może zawierać następujące znaki:

- Wielkie litery (A-Z)
- Małe litery (a-z)
- Cyfry (0-9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani wewnętrznych odstępów, ale może zawierać końcowe odstępy. Znak o kodzie zero może być używany do wskazania końca istotnych danych w nazwie; znak o kodzie zero i wszystkie następujące po nim znaki są traktowane jako odstępy. W podanych środowiskach obowiązują następujące ograniczenia:

- W systemach używających kodu EBCDIC Katakana nie można używać małych liter.
- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w cudzysłów, jeśli zostały podane w komendach. Te cudzysłowy nie mogą być podawane dla nazw, które występują jako pola w strukturach lub jako parametry w wywołaniach.

Do wskazanych typów obiektów mają zastosowanie następujące punkty:

- Jeśli *ODON* jest nazwą kolejki modelowej, menedżer kolejek tworzy kolejkę dynamiczną z atrybutami kolejki modelowej i zwraca w polu *ODON* nazwę utworzonej kolejki. Kolejkę modelową można określić tylko w wywołaniu MQOPEN; kolejka modelowa nie jest poprawna w wywołaniu MQPUT1 .
- Jeśli otwierany obiekt jest listą dystrybucyjną (*ODREC* jest obecny i większy od zera), *ODON* musi być pusty lub musi być łańcuchem pustym. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2152.
- Jeśli parametr *ODOT* ma wartość OTQM, mają zastosowanie reguły specjalne. W tym przypadku nazwa musi być całkowicie pusta aż do pierwszego znaku o kodzie zero lub końca pola.
- Jeśli ODON jest nazwą kolejki aliasowej z TARGTYPE (TOPIC), najpierw wykonywane jest sprawdzenie zabezpieczeń w nazwanej kolejce aliasowej, podobnie jak w przypadku kolejek aliasowych. Jeśli to sprawdzenie zabezpieczeń powiedzie się, wywołanie MQOPEN będzie

kontynuowane i będzie działać tak samo, jak wywołanie MQOPEN obiektu OTTOP, włącznie z przeprowadzeniem sprawdzenia zabezpieczeń względem obiektu tematu administracyjnego.

Jest to pole wejściowe/wyjściowe dla wywołania MQOPEN, gdy *ODON* jest nazwą kolejki modelowej, a we wszystkich innych przypadkach jest to pole tylko wejściowe. Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ODON* i *ODOS*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

ODORO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu obiektu od początku MQOD.

Jest to przesunięcie w bajtach pierwszego rekordu obiektu MQOR od początku struktury MQOD. Przesunięcie może być dodatnie lub ujemne. Opcja *ODORO* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *ODREC* ma wartość zero.

Podczas otwierania listy dystrybucyjnej należy podać tablicę zawierającą jeden lub więcej rekordów obiektu MQOR, aby określić nazwy kolejek docelowych na liście dystrybucyjnej. Można to zrobić na jeden z dwóch sposobów:

- Używając pola przesunięcia *ODORO*

  W takim przypadku aplikacja powinna zadeklarować własną strukturę zawierającą element MQOD, po którym następuje tablica rekordów MQOR (z wymaganą liczbą elementów tablicy) i ustawić wartość *ODORO* na przesunięcie pierwszego elementu tablicy od początku elementu MQOD. Należy upewnić się, że to przesunięcie jest poprawne.
- Za pomocą pola wskaźnika *ODORP*

  W takim przypadku aplikacja może zadeklarować tablicę struktur MQOR niezależnie od struktury MQOD i ustawić parametr *ODORP* na adres tablicy.

Niezależnie od wybranej techniki, należy użyć jednej z metod *ODORO* i *ODORP* ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2155 , jeśli obie są równe zero lub obie są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODORP (wskaźnik)

Adres pierwszego rekordu obiektu.

Jest to adres pierwszego rekordu obiektu MQOR. Opcja *ODORP* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *ODREC* ma wartość zero.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. Do określenia rekordów obiektów można użyć wartości *ODORP* lub *ODORO* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola *ODORO* . Jeśli parametr *ODORP* nie jest używany, musi być ustawiony na wskaźnik pusty lub bajty puste. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODOS (MQCHARV)

ODOS określa długą nazwę obiektu, która ma być użyta.

To pole jest przywoływane tylko w przypadku niektórych wartości pola *ODOT*. Szczegółowe informacje o tym, które wartości wskazują, że to pole jest używane, zawiera opis pola *ODOT* .

Jeśli parametr *ODOS* został podany niepoprawnie, zgodnie z opisem struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2441.

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ODON* i *ODOS*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER4.

ODOT (10-cyfrowa liczba całkowita ze znakiem)

Typ obiektu.

Typ obiektu, którego nazwa znajduje się w pliku *ODON*. Dozwolone są następujące wartości:

OTQ
do kolejki błędów. Nazwa obiektu znajduje się w katalogu *ODON*.

OTNLST
Lista nazw. Nazwa obiektu znajduje się w katalogu *ODON*.

OTPRO
Definicja procesu. Nazwa obiektu znajduje się w katalogu *ODON*.

OTQM,
menedżerze kolejek. Nazwa obiektu znajduje się w katalogu *ODON*.

OTTOP
. Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *ODON* i *ODOS*.

Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

Jeśli nie można znaleźć obiektu identyfikowanego przez pole *ODON* , wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2425 , nawet jeśli w parametrze *ODOS*podano łańcuch.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest OTQ.

ODREC (10-cyfrowa liczba całkowita ze znakiem)

Liczba rekordów obiektów.

Jest to liczba rekordów obiektów MQOR, które zostały udostępnione przez aplikację. Jeśli ta liczba jest większa od zera, oznacza to, że lista dystrybucyjna jest otwierana, a *ODREC* jest liczbą kolejek docelowych na liście. Poprawne jest, aby lista dystrybucyjna zawierała tylko jedno miejsce docelowe.

Wartość *ODREC* nie może być mniejsza niż zero, a jeśli jest większa niż zero *ODOT* musi być równa OTQ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2154 , jeśli te warunki nie są spełnione.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODRMN (48-bajtowy łańcuch znaków)

Rozstrzygnięta nazwa menedżera kolejek.

Jest to nazwa docelowego menedżera kolejek po wykonaniu tłumaczenia nazw przez lokalny menedżer kolejek. Zwracana nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez *ODRQN*. *ODRMN* może być nazwą lokalnego menedżera kolejek.

Jeśli *ODRQN* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejek, do której należy lokalny menedżer kolejek, *ODRMN* jest nazwą grupy współużytkowania kolejek. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, parametr *ODRQN* może być nazwą grupy współużytkowania kolejek lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejek (rodzaj zwracanej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą do przeglądania, wprowadzania lub wyprowadzania (lub dowolnej kombinacji). Jeśli otwierany jest dowolny z następujących obiektów, parametr *ODRMN* jest ustawiany na wartość pustą:

- To nie jest kolejka
- Kolejka, ale nie otwarta do przeglądania, wejścia lub wyjścia

- Kolejka klastra z określoną wartością OOBNDN (lub z wartością OOBNDQ, gdy atrybut kolejki **DefBind** ma wartość BNDNOT)
- Lista dystrybucyjna

Jest to pole wyjściowe. Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER3.

ODRO (MQCHARV)

ODRO jest długą nazwą obiektu po tym, jak menedżer kolejek rozstrzyga nazwę podaną w programie *ODON*.

To pole jest zwracane tylko dla określonych typów obiektów, tematów i aliasów kolejek, które odwołują się do obiektu tematu.

Jeśli w polu *ODOS* zostanie podana długa nazwa obiektu i w polu *ODON*nie zostanie podana żadna wartość, wartość zwrócona w tym polu będzie taka sama, jak w polu *ODOS*.

Jeśli to pole zostanie pominięte (to znaczy ODRO.VSBufSize ma wartość zero), parametr *ODRO* nie jest zwracany, ale długość jest zwracana w narzędziu ODRO.VSLength. Jeśli długość jest krótsza niż pełna *ODRO* , zostanie obcięta i zwróci tyle znaków po prawej stronie, ile może zmieścić się w podanej długości.

Jeśli parametr *ODRO* został podany niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2520. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER4.

ODRQN (48-bajtowy łańcuch znaków)

Nazwa rozstrzygniętej kolejki.

Jest to nazwa kolejki docelowej po wykonaniu tłumaczenia nazw przez menedżera kolejek lokalnych. Zwracana nazwa jest nazwą kolejki, która istnieje w menedżerze kolejek identyfikowanym przez *ODRMN*.

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą do przeglądania, wprowadzania lub wyprowadzania (lub dowolnej kombinacji). Jeśli otwierany jest dowolny z następujących obiektów, parametr *ODRQN* jest ustawiany na wartość pustą:

- To nie jest kolejka
- Kolejka, ale nie otwarta do przeglądania, wejścia lub wyjścia
- Lista dystrybucyjna
- Kolejka aliasowa, która odwołuje się do obiektu tematu (zamiast tego należy odwołać się do sekcji “ODRO (MQCHARV)” na stronie 1199)

Jest to pole wyjściowe. Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest łańcuch pusty w języku C i 48 znaków odstępu w innych językach programowania. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER3.

ODRRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu odpowiedzi od początku MQOD.

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQOD. Przesunięcie może być dodatnie lub ujemne. Opcja *ODRRO* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *ODREC* ma wartość zero.

Podczas otwierania listy dystrybucyjnej można podać tablicę zawierającą jeden lub więcej rekordów odpowiedzi MQRR w celu zidentyfikowania kolejek, których otwarcie nie powiodło się (pole*RRCC* w MQRR), oraz przyczyny każdego niepowodzenia (pole*RRREA* w MQRR). Dane są zwracane w tablicy rekordów odpowiedzi w tej samej kolejności, w jakiej nazwy kolejek występują w tablicy rekordów obiektów. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (niektóre kolejki zostały pomyślnie otwarte, podczas gdy inne nie powiodły się lub wszystkie zakończyły się niepowodzeniem, ale z różnych powodów). Kod przyczyny RC2136 z wywołania

wskazuje ten przypadek. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna jest zwracana w parametrze **REASON** wywołania MQOPEN lub MQPUT1 , a rekordy odpowiedzi nie są ustawione. Rekordy odpowiedzi są opcjonalne, ale jeśli zostaną podane, musi być ich *ODREC* .

Rekordy odpowiedzi mogą być udostępniane w taki sam sposób, jak rekordy obiektów, przez określenie przesunięcia w *ODRRO*lub przez określenie adresu w *ODRRP* ; Szczegółowe informacje na ten temat można znaleźć w opisie komendy *ODORO* . Można jednak użyć nie więcej niż jednej z wartości *ODRRO* i *ODRRP* ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2156 , jeśli obie wartości są niezerowe.

W przypadku wywołania MQPUT1 te rekordy odpowiedzi są używane do zwracania informacji o błędach, które wystąpiły podczas wysyłania komunikatu do kolejek na liście dystrybucyjnej, a także o błędach, które wystąpiły podczas otwierania kolejek. Kod zakończenia i kod przyczyny z operacji umieszczania dla kolejki zastępują te z operacji otwierania dla tej kolejki tylko wtedy, gdy kod zakończenia z tej ostatniej to CCOK lub CCWARN.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODRRP (wskaźnik)

Adres pierwszego rekordu odpowiedzi.

Jest to adres pierwszego rekordu odpowiedzi MQRR. Opcja *ODRRP* jest używana tylko wtedy, gdy otwierana jest lista dystrybucyjna. Pole jest ignorowane, jeśli parametr *ODREC* ma wartość zero.

Do określenia rekordów odpowiedzi można użyć wartości *ODRRP* lub *ODRRO* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w poprzednim opisie pola *ODRRO* . Jeśli parametr *ODRRP* nie jest używany, musi być ustawiony na wskaźnik pusty lub bajty puste.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

ODSIDV
Identyfikator struktury deskryptora obiektu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest ODSIDV.

ODSS (MQCHARV)

ODSS zawiera łańcuch określający kryteria wyboru używane podczas pobierania komunikatów z kolejki.

Parametr *ODSS* nie może być podany w następujących przypadkach:

- Jeśli *ODOT* nie jest równe OTQ
- Jeśli otwierana kolejka nie jest otwierana przy użyciu jednej z opcji wejściowych, OOINP*

Jeśli w takich przypadkach zostanie podana wartość *ODSS* , wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2516.

Jeśli parametr *ODSS* został podany niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2519. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER4.

ODUDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba pomyślnie otwartych kolejek zdalnych

Jest to liczba kolejek na liście dystrybucyjnej, które są tłumaczone na kolejki zdalne i które zostały pomyślnie otwarte. Jeśli istnieje, to pole jest również ustawiane podczas otwierania pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *ODVER* jest mniejsza niż ODVER2.

ODVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

ODVER1
Struktura deskryptora obiektu Version-1 .

ODVER2
Struktura deskryptora obiektu Version-2 .

ODVER3
Struktura deskryptora obiektu Version-3 .

ODVER4
Struktura deskryptora obiektu Version-4 .

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

ODVERC
Bieżąca wersja struktury deskryptora obiektu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest ODVER1.

Wartości początkowe

Tabela 713. Pola w MQOD

Nazwa pola	Nazwa stałej	Wartość stałej
ODSID	ODSIDV	'OD¬¬'
ODVER	ODVER1	1
ODOT	OTQ	1
ODON	Brak	Puste
ODMN	Brak	Puste
ODDN	Brak	'AMQ.*'
ODAU	Brak	Puste
ODREC	Brak	0
ODKDC	Brak	0
ODUDC	Brak	0
ODIDC	Brak	0
ODORO	Brak	0
ODRRO	Brak	0
ODORP	Brak	Pusty wskaźnik lub puste bajty
ODRRP	Brak	Pusty wskaźnik lub puste bajty
ODASI	SINON	Wartości null

Tabela 713. Pola w MQOD (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
ODRQN	Brak	Puste
ODRMN	Brak	Puste
ODOS	Zgodnie z definicją dla MQCHARV	Zgodnie z definicją dla MQCHARV
ODRO	Zgodnie z opisem w sekcji *ODOS*	Zgodnie z opisem w sekcji *ODOS*
ODSS	Brak	Puste
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQOD Structure
D*
D*
D* Structure identifier
D  ODSID                  1      4    INZ('OD  ')
D*
D* Structure version number
D  ODVER                  5      8I 0 INZ(1)
D*
D* Object type
D  ODOT                   9     12I 0 INZ(1)
D*
D* Object name
D  ODON                  13     60    INZ
D*
D* Object queue manager name
D  ODMN                  61    108    INZ
D*
D* Dynamic queue name
D  ODDN                 109    156    INZ('AMQ.*')
D*
D* Alternate user identifier
D  ODAU                 157    168    INZ
D*
** Number of object records
D* present
D  ODREC                169    172I 0 INZ(0)
D*
** Number of local queues opened
D* successfully
D  ODKDC                173    176I 0 INZ(0)
D*
** Number of remote queues opened
D* successfully
D  ODUDC                177    180I 0 INZ(0)
D*
** Number of queues that failed to
D* open
D  ODIDC                181    184I 0 INZ(0)
D*
** Offset of first object record
D* from start of MQOD
D  ODORO                185    188I 0 INZ(0)
D*
** Offset of first response record
D* from start of MQOD
D  ODRRO                189    192I 0 INZ(0)
D*
D* Address of first object record
D  ODORP                193    208*   INZ(*NULL)
```

```
     D*
     ** Address of first response
     D* record
     D  ODRRP                209    224*   INZ(*NULL)
     D*
     D* Alternate security identifier
     D  ODASI                225    264    INZ(X'00000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000000000000000-
     D                                     00000000000000')
     D*
     D* Resolved queue name
     D  ODRQN                265    312    INZ
     D*
     D* Resolved queue manager name
     D  ODRMN                313    360    INZ
     D*
     D* reserved field
     D  ODRE1                361    364I 0 INZ(0)
     D*
     D* reserved field
     D  ODRS2                365    368I 0 INZ(0)
     D*
     D* Object long name
     D* Address of variable length string
     D  ODOSCHRP             369    384*   INZ(*NULL)
     D* Offset of variable length string
     D  ODOSCHRO             385    388I 0 INZ(0)
     D* Size of buffer
     D  ODOSVSBS             389    392I 0 INZ(-1)
     D* Length of variable length string
     D  ODOSCHRL             393    396I 0 INZ(0)
     D* CCSID of variable length string
     D  ODOSCHRC             397    400I 0 INZ(-3)
     D*
     D* Message Selector
     D* Address of variable length string
     D  ODSSCHRP             401    416*   INZ(*NULL)
     D* Offset of variable length string
     D  ODSSCHRO             417    420I 0 INZ(0)
     D* Size of buffer
     D  ODSSVSBS             421    424I 0 INZ(-1)
     D* Length of variable length string
     D  ODSSCHRL             425    428I 0 INZ(0)
     D* CCSID of variable length string
     D  ODSSCHRC             429    432I 0 INZ(-3)
     D*
     D* Resolved long object name
     D* Address of variable length string
     D  ODRSOCHRP            433    448*   INZ(*NULL)
     D* Offset of variable length string
     D  ODRSOCHRO            449    452I 0 INZ(0)
     D* Size of buffer
     D  ODRSOVSBS            453    456I 0 INZ(-1)
     D* Length of variable length string
     D  ODRSOCHRL            457    460I 0 INZ(0)
     D* CCSID of variable length string
     D  ODRSOCHRC            461    464I 0 INZ(-3)
     D*
     D* Alias queue resolved object type
     D  ODRT                 465    468I 0 INZ(0)
```

IBM i MQOR (rekord obiektu) w systemie IBM i

Struktura MQOR służy do określania nazwy kolejki i nazwy menedżera kolejek pojedynczej kolejki docelowej.

Przegląd

Cel: MQOR jest strukturą wejściową dla wywołań MQOPEN i MQPUT1 .

Zestaw znaków i kodowanie: Dane w MQOR muszą być w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Użycie: Dzięki udostępnieniu tablicy tych struktur w wywołaniu MQOPEN można otworzyć listę kolejek. Lista ta jest nazywana *listą dystrybucyjną*. Jeśli kolejka została pomyślnie otwarta, każdy komunikat umieszczony za pomocą uchwytu kolejki zwróconego przez wywołanie MQOPEN jest umieszczany w każdej z kolejek na liście.

- "Pola" na stronie 1204
- "Wartości początkowe" na stronie 1204
- "Deklaracja RPG" na stronie 1204

Pola

Struktura MQOR zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

ORMN (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek obiektu.

Jest to takie samo jak pole *ODMN* w strukturze MQOD (szczegółowe informacje zawiera sekcja MQOD).

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest 48 znaków odstępu.

ORON (48-bajtowy łańcuch znaków)

Nazwa obiektu.

Jest to to samo pole, co pole *ODON* w strukturze MQOD (szczegółowe informacje zawiera dokument MQOD), z tą różnicą, że:

- Musi to być nazwa kolejki.
- Nie może to być nazwa kolejki modelowej.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest 48 znaków odstępu.

Wartości początkowe

Tabela 714. Pola w MQOR

Nazwa pola	Nazwa stałej	Wartość stałej
ORON	Brak	Puste
ORMN	Brak	Puste

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQOR Structure
D*
D* Object name
D  ORON                  1     48    INZ
D* Object queue manager name
D  ORMN                 49     96    INZ
```

MQPD-deskryptor właściwości

MQPD służy do definiowania atrybutów właściwości.

Przegląd

Cel: Struktura jest parametrem wejścia/wyjścia w wywołaniu MQSETMP i parametrem wyjściowym w wywołaniu MQINQMP.

Zestaw znaków i kodowanie: dane w MQPD muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1205
- "Wartości początkowe" na stronie 1207
- "Deklaracja RPG" na stronie 1207

Pola

Struktura MQPD zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

PDCT (10-cyfrowa liczba całkowita ze znakiem)

Opisuje kontekst komunikatu, do którego należy właściwość.

Gdy menedżer kolejek odbierze komunikat zawierający właściwość zdefiniowaną przez IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną. Menedżer kolejek poprawi wartość pola *PDCT* .

Można podać następującą opcję:

PDC
Właściwość jest powiązana z kontekstem użytkownika.

Aby można było ustawić właściwość powiązaną z kontekstem użytkownika za pomocą wywołania MQSETMP, nie jest wymagana żadna specjalna autoryzacja.

Jeśli opisana wcześniej opcja nie jest wymagana, można użyć następującej opcji:

PDC
Właściwość nie jest powiązana z kontekstem komunikatu.

Nierozpoznana wartość jest odrzucana z kodem *PDREA* RC2482.

Jest to pole wejściowe/wyjściowe wywołania MQSETMP i pole wyjściowe wywołania MQINQMP. Wartością początkową tego pola jest PDNOC.

PDCPYOPT (10-cyfrowa liczba całkowita ze znakiem)

Opisuje typ komunikatów, do których właściwość powinna zostać skopiowana.

Jest to pole wyjściowe tylko dla rozpoznawanych właściwości zdefiniowanych przez IBM MQ; IBM MQ ustawia odpowiednią wartość.

Gdy menedżer kolejek odbierze komunikat zawierający właściwość zdefiniowaną przez IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną. Menedżer kolejek poprawi wartość pola *CopyOptions* .

Można określić jedną lub więcej z tych opcji. Aby określić więcej niż jedną opcję, dodaj wartości razem (nie dodawaj więcej niż raz tej samej stałej) lub połącz wartości za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

POKOP
Ta właściwość jest kopiowana do przekazywanej wiadomości.

POPUB
Ta właściwość jest kopiowana do komunikatu odebranego przez subskrybenta podczas publikowania komunikatu.

KOPREP
Ta właściwość jest kopiowana do komunikatu odpowiedzi.

KOPRP
Ta właściwość jest kopiowana do komunikatu raportu.

KOPIUJE
Ta właściwość jest kopiowana do wszystkich typów kolejnych komunikatów.

COPNON
Ta właściwość nie jest kopiowana do komunikatu.

Opcja domyślna: Można podać następującą opcję, aby określić domyślny zestaw opcji kopiowania:

POChP
Ta właściwość jest kopiowana do przekazywanego komunikatu, do komunikatu raportu lub do komunikatu odebranego przez subskrybent podczas publikowania komunikatu.

Jest to równoważne z określeniem kombinacji opcji COPFOR, COPRP i COPPUB.

Jeśli żadna z opisanych wcześniej opcji nie jest wymagana, należy użyć następującej opcji:

COPNON
Ta wartość wskazuje, że nie określono żadnych innych opcji kopiowania; programowo nie istnieje relacja między tą właściwością a kolejnymi komunikatami. Ta wartość jest zawsze zwracana dla właściwości deskryptora komunikatu.

Jest to pole wejściowe/wyjściowe wywołania MQSETMP i pole wyjściowe wywołania MQINQMP. Wartością początkową tego pola jest COPDEF.

PDOPT (10-cyfrowa liczba całkowita ze znakiem)

Wartość musi być następująca:

BRAK PDNONE
Nie określono żadnych opcji

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest PDNONE.

PDSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury; wartość musi być następująca:

PPSIDV
Identyfikator struktury deskryptora właściwości.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **PSIDV**.

PDSUP (10-cyfrowa liczba całkowita ze znakiem)

W tym polu opisano, jaki poziom obsługi właściwości komunikatu jest wymagany przez menedżer kolejek, aby komunikat zawierający tę właściwość został umieszczony w kolejce. Dotyczy to tylko właściwości zdefiniowanych przez IBM MQ; obsługa wszystkich innych właściwości jest opcjonalna.

Pole jest automatycznie ustawiane na poprawną wartość, gdy właściwość zdefiniowana przez program IBM MQjest znana menedżerowi kolejek. Jeśli właściwość nie zostanie rozpoznana, zostanie przypisana wartość PDSUPO. Gdy menedżer kolejek odbierze komunikat zawierający właściwość zdefiniowaną przez IBM MQ, którą menedżer kolejek rozpoznaje jako niepoprawną. Menedżer kolejek poprawi wartość pola *PDSUP* .

Podczas ustawiania właściwości zdefiniowanej przez IBM MQza pomocą wywołania MQSETMP dla uchwytu komunikatu, w którym została ustawiona opcja CMNOVA, *PDSUP* staje się polem wejściowym. Dzięki temu aplikacja może umieścić właściwość zdefiniowaną w programie IBM MQo poprawnej wartości, jeśli właściwość nie jest obsługiwana przez połączony menedżer kolejek, ale komunikat ma być przetwarzany w innym menedżerze kolejek.

Wartość PDSUPO jest zawsze przypisywana do właściwości, które nie są właściwościami zdefiniowanymi przez IBM MQ.

Jedna z następujących wartości jest zwracana przez wywołanie MQINQMP lub jedna z wartości może zostać określona, jeśli jest używane wywołanie MQSETMP dla uchwytu komunikatu, w którym ustawiono opcję CMNOVA:

PDSUPO
Właściwość jest akceptowana przez menedżer kolejek, nawet jeśli nie jest obsługiwana. Właściwość można odrzucić, aby komunikat przepływał do menedżera kolejek, który nie obsługuje właściwości komunikatu. Ta wartość jest również przypisywana do właściwości, które nie są zdefiniowane w systemie IBM MQ.

PSUPR
Obsługa tej właściwości jest wymagana. Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje właściwości zdefiniowanej przez IBM MQ. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2490.

PSUPL
Komunikat jest odrzucany przez menedżer kolejek, który nie obsługuje właściwości zdefiniowanej przez IBM MQ, jeśli komunikat jest przeznaczony dla kolejki lokalnej. Wywołanie MQPUT lub MQPUT1 kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2490.

Wywołanie MQPUT lub MQPUT1 powiedzie się, jeśli komunikat jest przeznaczony dla zdalnego menedżera kolejek.

Jest to pole wyjściowe wywołania MQINQMP i pole wejściowe wywołania MQSETMP, jeśli uchwyt komunikatu został utworzony z ustawioną opcją CMNOVA. Wartością początkową tego pola jest PDSUPO.

PDVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

PDVER1
Struktura deskryptora właściwości Version-1 .

Następująca stała określa numer wersji bieżącej:

PDVERC (PDVERC)
Bieżąca wersja struktury deskryptora właściwości.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **PDVER1**.

Wartości początkowe

Tabela 715. Pola w MQPD

Nazwa pola	Nazwa stałej	Wartość stałej
PDSID	PDSIDV	'PD'
PDVER	PDVER1	1
PDOPT	BRAK PDNONE	0
PDSUP	PDSUPO	0
PDCT	PDC	0
PDCPYOPT	POChP	0

Deklaracja RPG

```
D* MQDMHO Structure
D*
D*
D* Structure identifier
D  DMSID                  1      4    INZ('DMHO')
D*
D* Structure version number
D  DMVER                  5      8I 0 INZ(1)
D*
D* Options that control the action of MQDLTMH
D  DMOPT                  9     12I 0 INZ(0)
```

IBM i MQPMO (opcje umieszczania komunikatów) w systemie IBM i

Struktura MQPMO umożliwia aplikacji określanie opcji sterujących umieszczaniem komunikatów w kolejkach lub publikowaniem ich w tematach.

Przegląd

Przeznaczenie
Struktura jest parametrem wejścia/wyjścia w wywołaniach MQPUT i MQPUT1 .

Wersja
Bieżąca wersja produktu MQPMO to PMVER2. Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w kolejnych opisach.

Udostępniony plik COPY zawiera najnowszą wersję programu MQPMO, która jest obsługiwana przez środowisko, ale z wartością początkową pola *PMVER* ustawioną na PMVER1. Aby użyć pól, które nie są obecne w strukturze version-1 , aplikacja musi ustawić w polu *PMVER* numer wersji wymaganej wersji.

Zestaw znaków i kodowanie
Dane w MQPMO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

- "Pola" na stronie 1208
- "Wartości początkowe" na stronie 1223
- "Deklaracja RPG" na stronie 1223

Pola

Struktura MQPMO zawiera następujące pola. Pola są opisane w porządku alfabetycznym:

PMCT (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt obiektu kolejki wejściowej.

Jeśli określono PMPASI lub PMPASA, to pole musi zawierać uchwyt kolejki wejściowej, z którego pobierane są informacje o kontekście, które mają być powiązane z umieszczanym komunikatem.

Jeśli PMPASI i PMPASA nie są określone, to pole jest ignorowane.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

PMIDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba komunikatów, których nie można było wysłać.

Jest to liczba komunikatów, których nie można było wysłać do kolejek na liście dystrybucyjnej. Liczba ta obejmuje kolejki, których otwarcie nie powiodło się, oraz kolejki, które zostały pomyślnie otwarte, ale dla których operacja umieszczania nie powiodła się. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Uwaga: To pole jest ustawiane tylko wtedy, gdy parametr **CMPCOD** w wywołaniu MQPUT lub MQPUT1 ma wartość CCOK lub CCWARN. Nie jest ustawiane, jeśli parametr **CMPCOD** ma wartość CCFAIL.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawiane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMKDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba komunikatów pomyślnie wysłanych do kolejek lokalnych.

Jest to liczba komunikatów, które zostały pomyślnie wysłane przez bieżące wywołania MQPUT lub MQPUT1 do kolejek znajdujących się na liście dystrybucyjnej, które są kolejkami lokalnymi. Licznik nie uwzględnia komunikatów wysłanych do kolejek, które są tłumaczone na kolejki zdalne (nawet jeśli początkowo do przechowywania komunikatu używana jest lokalna kolejka transmisji). To pole jest

również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawiane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje sterujące działaniem komend MQPUT i MQPUT1.

Można określić dowolną z poniższych wartości lub nie można określić żadnej z nich. Jeśli wymagana jest więcej niż jedna wartość, można dodać wartości (nie należy dodawać tej samej stałej więcej niż raz). Kombinacje, które nie są poprawne, są odnotowywane; wszystkie inne kombinacje są poprawne.

Opcje publikowania: Następujące opcje sterują sposobem publikowania komunikatów w temacie.

PMSRTO

Żadne informacje wypełnione w polach MDRQ i MDRM deskryptora MQMD tej publikacji nie są przekazywane do subskrybentów. Jeśli ta opcja jest używana z opcją raportu, która wymaga odpowiedzi ReplyToQ, wywołanie kończy się niepowodzeniem z błędem RC2027 .

PMRET

Wysyłana publikacja ma zostać zachowana przez menedżer kolejek. Umożliwia to subskrybentowi żądanie kopii tej publikacji po jej opublikowaniu za pomocą wywołania MQSUBRQ. Umożliwia również wysyłanie publikacji do aplikacji, które dokonały subskrypcji po tym, jak publikacja ta została dokonana, chyba że publikacja nie zostanie wysłana przy użyciu opcji SONEWP. Jeśli do aplikacji zostanie wysłana zachowana publikacja, jest ona wskazywana przez właściwość komunikatu mq.IsRetained tej publikacji.

W każdym węźle drzewa tematów może być zachowana tylko jedna publikacja. Oznacza to, że jeśli istnieje już zachowana publikacja dla tego tematu, opublikowana przez dowolną inną aplikację, zostanie ona zastąpiona tą publikacją. Dlatego lepiej jest unikać sytuacji, w której więcej niż jeden publikator zachowuje komunikaty w tym samym temacie.

Jeśli subskrybent żąda zachowanych publikacji, używana subskrypcja może zawierać znak wieloznaczny w temacie. W takim przypadku pewna liczba zachowanych publikacji może być zgodna (w różnych węzłach drzewa tematów) i kilka publikacji może zostać wysłanych do aplikacji żądającej. Więcej szczegółów zawiera opis wywołania “MQSUBRQ-żądanie subskrypcji” na stronie 821 .

Jeśli ta opcja jest używana i nie można zachować publikacji, komunikat nie jest publikowany i wywołanie kończy się niepowodzeniem z komunikatem RC2479 .

Opcje punktu synchronizacji: Następujące opcje odnoszą się do udziału wywołania MQPUT lub MQPUT1 w jednostce pracy:

PMSYP

Umieść komunikat z elementem sterującym punktu synchronizacji.

Żądanie ma działać w ramach zwykłych protokołów jednostki pracy. Komunikat nie jest widoczny poza jednostką pracy, dopóki jednostka pracy nie zostanie zatwierdzona. Jeśli jednostka pracy zostanie wycofana, komunikat zostanie usunięty.

Jeśli ta opcja i PMNSYP nie są określone, żądanie umieszczenia nie znajduje się w jednostce pracy.

Parametr PMSYP nie może być określony z parametrem PMNSYP.

PMNSYP,

Umieść komunikat bez elementu sterującego punktu synchronizacji.

Żądanie ma działać poza normalnymi protokołami jednostki pracy. Komunikat jest dostępny natychmiast i nie można go usunąć przez wycofanie jednostki pracy.

Jeśli ta opcja i PMSYP nie są określone, żądanie umieszczenia nie znajduje się w jednostce pracy.

Parametr PMNSYP nie może być określony z parametrem PMSYP.

Opcje identyfikatora komunikatu i identyfikatora korelacji: Następujące opcje żądają od menedżera kolejek wygenerowania nowego identyfikatora komunikatu lub identyfikatora korelacji:

PMNMID,
Wygeneruj nowy identyfikator komunikatu.

Ta opcja powoduje, że menedżer kolejek zastępuje treść pola *MDMID* w strukturze MQMD nowym identyfikatorem komunikatu. Ten identyfikator komunikatu jest wysyłany wraz z komunikatem i zwracany do aplikacji w wyniku wywołania MQPUT lub MQPUT1 .

Tę opcję można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej. Szczegółowe informacje można znaleźć w opisie pola *PRMID* w strukturze MQPMR.

Użycie tej opcji zmniejsza konieczność zresetowania pola *MDMID* na wartość MINONE przed każdym wywołaniem MQPUT lub MQPUT1 .

Identyfikator PMNCID
Wygeneruj nowy identyfikator korelacji.

Ta opcja powoduje, że menedżer kolejek zastępuje zawartość pola *MDCID* w strukturze MQMD nowym identyfikatorem korelacji. Ten identyfikator korelacji jest wysyłany z komunikatem i zwracany do aplikacji w wyniku wywołania MQPUT lub MQPUT1 .

Tę opcję można również określić, gdy komunikat jest umieszczany na liście dystrybucyjnej. Szczegółowe informacje można znaleźć w opisie pola *PRCID* w strukturze MQPMR.

Identyfikator PMNCID jest przydatny w sytuacjach, w których aplikacja wymaga unikalnego identyfikatora korelacji.

Opcje grup i segmentów: Poniższa opcja odnosi się do przetwarzania komunikatów w grupach i segmentach komunikatów logicznych. Poniższe definicje mogą pomóc w zrozumieniu tej opcji:

Komunikat fizyczny
Jest to najmniejsza jednostka informacji, którą można umieścić w kolejce lub z niej usunąć. Często odpowiada ona informacjom określonym lub pobranym w pojedynczym wywołaniu MQPUT, MQPUT1lub MQGET. Każdy komunikat fizyczny ma własny deskryptor komunikatu (MQMD). Zazwyczaj komunikaty fizyczne są rozróżniane na podstawie różnych wartości identyfikatora komunikatu (pole*MDMID* w strukturze MQMD), chociaż nie jest to wymuszane przez menedżer kolejek.

Komunikat logiczny
Jest to pojedyncza jednostka informacji o aplikacji. W przypadku braku ograniczeń systemowych komunikat logiczny byłby taki sam, jak komunikat fizyczny. Jeśli jednak komunikaty logiczne są duże, ograniczenia systemowe mogą spowodować, że wskazane lub konieczne będzie podzielenie komunikatu logicznego na dwa lub więcej komunikatów fizycznych, nazywanych *segmentami*.

Komunikat logiczny, który został podzielony na segmenty, składa się z dwóch lub większej liczby komunikatów fizycznych, które mają ten sam identyfikator grupy o wartości innej niż NULL (pole*MDGID* w strukturze MQMD) i ten sam numer kolejny komunikatu (pole*MDSEQ* w strukturze MQMD). Segmenty są rozróżniane przez różne wartości przesunięcia segmentu (pole*MDOFF* w strukturze MQMD), które powoduje przesunięcie danych w komunikacie fizycznym od początku danych w komunikacie logicznym. Ponieważ każdy segment jest komunikatem fizycznym, segmenty w komunikacie logicznym zwykle mają różne identyfikatory komunikatów.

Komunikat logiczny, który nie został posegmentowany, ale dla którego segmentacja została dozwolona przez aplikację wysyłającą, ma również identyfikator grupy o wartości innej niż NULL, chociaż w tym przypadku istnieje tylko jeden komunikat fizyczny z tym identyfikatorem grupy, jeśli komunikat logiczny nie należy do grupy komunikatów. Komunikaty logiczne, dla których segmentacja została zablokowana przez aplikację wysyłającą, mają pusty identyfikator grupy (GINONE), chyba że komunikat logiczny należy do grupy komunikatów.

Wyślij wiadomość do grupy
Jest to zestaw co najmniej jednego komunikatu logicznego, który ma ten sam identyfikator grupy o wartości innej niż NULL. Komunikaty logiczne w grupie są rozróżniane przez różne wartości numeru kolejnego komunikatu, który jest liczbą całkowitą z zakresu od 1 do n, gdzie n jest

liczbą komunikatów logicznych w grupie. Jeśli jeden lub więcej komunikatów logicznych jest segmentowanych, w grupie znajduje się więcej niż n komunikatów fizycznych.

PMLOGO
Komunikaty w grupach i segmentach komunikatów logicznych są umieszczane w porządku logicznym.

Ta opcja informuje menedżera kolejek, w jaki sposób aplikacja umieszcza komunikaty w grupach i segmentach komunikatów logicznych. Można go określić tylko w wywołaniu MQPUT. Nie jest on poprawny w wywołaniu MQPUT1 .

Jeśli określono PMLOGO, oznacza to, że aplikacja używa kolejnych wywołań MQPUT do:

- Umieszczenie segmentów w każdym komunikacie logicznym w kolejności rosnącego przesunięcia segmentu, począwszy od 0, bez przerw.
- Przed umieszczeniem segmentów w następnym komunikacie logicznym należy umieścić wszystkie segmenty w jednym komunikacie logicznym.
- Umieszczenie komunikatów logicznych w każdej grupie komunikatów w kolejności rosnących numerów kolejnych komunikatów, począwszy od 1, bez przerw.
- Przed umieszczeniem komunikatów logicznych w następnej grupie komunikatów umieść wszystkie komunikaty logiczne w jednej grupie komunikatów.

Ta kolejność jest nazywana "kolejnością logiczną".

Ponieważ aplikacja powiedziała menedżerowi kolejek, w jaki sposób umieszcza komunikaty w grupach i segmentach komunikatów logicznych, nie musi ona obsługiwać i aktualizować informacji o grupach i segmentach dla każdego wywołania MQPUT, tak jak robi to menedżer kolejek. W szczególności oznacza to, że aplikacja nie musi ustawiać pól *MDGID*, *MDSEQ*i *MDOFF* w strukturze MQMD, ponieważ menedżer kolejek ustawia je na odpowiednie wartości. Aplikacja musi ustawić tylko pole *MDMFL* w strukturze MQMD, aby wskazać, kiedy komunikaty należą do grup lub są segmentami komunikatów logicznych, oraz aby wskazać ostatni komunikat w grupie lub ostatni segment komunikatu logicznego.

Po uruchomieniu grupy komunikatów lub komunikatu logicznego kolejne wywołania MQPUT muszą określać odpowiednie flagi MF* w pliku *MDMFL* w deskryptorze MQMD. Jeśli aplikacja próbuje umieścić komunikat nie w grupie, gdy istnieje niezakończona grupa komunikatów, lub umieścić komunikat, który nie jest segmentem, gdy istnieje niezakończony komunikat logiczny, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2241 lub RC2242 (w zależności od przypadku). Jednak menedżer kolejek zachowuje informacje o bieżącej grupie komunikatów lub bieżącym komunikacie logicznym, a aplikacja może je zakończyć, wysyłając komunikat (być może bez danych komunikatu aplikacji), określając odpowiednio MFLMIG lub MFLSEG przed ponownym wywołaniem MQPUT w celu umieszczenia komunikatu, który nie znajduje się w grupie lub nie jest segmentem.

Tabela 716 na stronie 1212 przedstawia kombinacje poprawnych opcji i flag oraz wartości pól *MDGID*, *MDSEQ*i *MDOFF* używane przez menedżer kolejek w każdym przypadku. Kombinacje opcji i flag, które nie są wyświetlane w tabeli, są niepoprawne. Kolumny w tabeli mają następujące znaczenie:

SŁOWO PROTOKOŁU
Wskazuje, czy opcja PMLOGO jest określona w wywołaniu.

MIG
Wskazuje, czy opcja MFMIG lub MFLMIG jest określona w wywołaniu.

SEG
Wskazuje, czy opcja MFSEG lub MFLSEG jest określona w wywołaniu.

SEG OK
Wskazuje, czy w wywołaniu określono opcję MFSEGA.

Cur grp
Wskazuje, czy przed wywołaniem istnieje bieżąca grupa komunikatów.

Komunikat dziennika Cur
Wskazuje, czy przed wywołaniem istnieje bieżący komunikat logiczny.

Inne kolumny
Pokaż wartości używane przez menedżer kolejek. "Poprzedni" oznacza wartość używaną dla pola w poprzednim komunikacie dla uchwytu kolejki.

PMRLOC
Określa, że nazwa PMRQN w strukturze MQPMO musi być zakończona nazwą kolejki lokalnej, do której komunikat jest rzeczywiście umieszczany. Nazwa ResolvedQMgrjest podobnie wypełniona nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną. Patrz OORLOQ, aby dowiedzieć się, co to oznacza. Jeśli użytkownik ma uprawnienia do umieszczania w kolejce, to ma uprawnienia wymagane do określenia tej flagi w wywołaniu MQPUT. Nie są wymagane żadne uprawnienia specjalne.

Tabela 716. Opcje MQPUT dotyczące komunikatów w grupach i segmentach komunikatów logicznych

Określone opcje				**Status grupy i komunikatu dziennika przed wywołaniem**		**Wartości używane przez menedżer kolejek**		
SŁOWO PROTOKOŁU	**MIG (MIG)**	**SEG (SEG)**	**PATRZ POPRAWNIE**	**Bieżąca grupa**	**Bieżące komunikaty dziennika**	*MDGID*	*MDSEQ*	*MDOFF*
Tak	Nie	Nie	Nie	Nie	Nie	GINON	1	0
Tak	Nie	Nie	Tak	Nie	Nie	Nowy identyfikator grupy	1	0
Tak	Nie	Tak	Tak lub Nie	Nie	Nie	Nowy identyfikator grupy	1	0
Tak	Nie	Tak	Tak lub Nie	Nie	Tak	Poprzedni identyfikator grupy	1	Poprzednie przesunięcie + poprzednia długość segmentu
Tak	Tak	Tak lub Nie	Tak lub Nie	Nie	Nie	Nowy identyfikator grupy	1	0
Tak	Tak	Tak lub Nie	Tak lub Nie	Tak	Nie	Poprzedni identyfikator grupy	Poprzedni numer kolejny + 1	0
Tak	Tak	Tak	Tak lub Nie	Tak	Tak	Poprzedni identyfikator grupy	Poprzedni numer kolejny	Poprzednie przesunięcie + poprzednia długość segmentu
Nie	Nie	Nie	Nie	Tak lub Nie	Tak lub Nie	GINON	1	0

Tabela 716. Opcje MQPUT dotyczące komunikatów w grupach i segmentach komunikatów logicznych (kontynuacja)

Określone opcje				Status grupy i komunikatu dziennika przed wywołaniem		Wartości używane przez menedżer kolejek		
Nie	Nie	Nie	Tak	Tak lub Nie	Tak lub Nie	Nowy identyfikator grupy, jeśli GINONE, w przeciwnym razie wartość w polu	1	0
Nie	Nie	Tak	Tak lub Nie	Tak lub Nie	Tak lub Nie	Nowy identyfikator grupy, jeśli GINONE, w przeciwnym razie wartość w polu	1	Wartość w polu
Nie	Tak	Nie	Tak lub Nie	Tak lub Nie	Tak lub Nie	Nowy identyfikator grupy, jeśli GINONE, w przeciwnym razie wartość w polu	Wartość w polu	0
Nie	Tak	Tak	Tak lub Nie	Tak lub Nie	Tak lub Nie	Nowy identyfikator grupy, jeśli GINONE, w przeciwnym razie wartość w polu	Wartość w polu	Wartość w polu

Uwaga:

- PMLOGO nie jest poprawne w wywołaniu MQPUT1 .
- W przypadku pola *MDMID* menedżer kolejek generuje nowy identyfikator komunikatu, jeśli określono wartość PMNMID lub MINONE, i w przeciwnym razie używa wartości z pola.
- W przypadku pola *MDCID* menedżer kolejek generuje nowy identyfikator korelacji, jeśli określono identyfikator PMNCID, i w przeciwnym razie używa wartości z tego pola.

Jeśli określono PMLOGO, menedżer kolejek wymaga, aby wszystkie komunikaty w grupie i segmenty w komunikacie logicznym były umieszczane z taką samą wartością w polu *MDPER* w deskryptorze MQMD, to znaczy, aby wszystkie komunikaty były trwałe lub aby wszystkie były nietrwałe. Jeśli ten warunek nie jest spełniony, wywołanie MQPUT kończy się niepowodzeniem z kodem przyczyny RC2185 .

Opcja PMLOGO wpływa na jednostki pracy w następujący sposób:

- Jeśli pierwszy komunikat fizyczny w grupie lub komunikat logiczny jest umieszczany w jednostce pracy, wszystkie inne komunikaty fizyczne w grupie lub komunikat logiczny muszą być umieszczane w jednostce pracy, jeśli używany jest ten sam uchwyt kolejki. Nie muszą one jednak znajdować się w tej samej jednostce pracy. Umożliwia to podział grupy komunikatów lub komunikatu logicznego składającego się z wielu komunikatów fizycznych na dwie lub więcej kolejnych jednostek pracy dla uchwytu kolejki.

- Jeśli pierwszy komunikat fizyczny w grupie lub komunikacie logicznym nie zostanie umieszczony w jednostce pracy, żaden inny komunikat fizyczny w grupie lub komunikacie logicznym nie może zostać umieszczony w jednostce pracy, jeśli używany jest ten sam uchwyt kolejki.

Jeśli te warunki nie są spełnione, wywołanie MQPUT kończy się niepowodzeniem z kodem przyczyny RC2245 .

Jeśli określono PMLOGO, wartość MQMD podana w wywołaniu MQPUT nie może być mniejsza niż MDVER2. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2257 .

Jeśli nie określono PMLOGO, komunikaty w grupach i segmentach komunikatów logicznych mogą być umieszczane w dowolnej kolejności i nie jest konieczne umieszczanie kompletnych grup komunikatów ani kompletnych komunikatów logicznych. Odpowiedzialność za to, aby pola *MDGID*, *MDSEQ*, *MDOFF*i *MDMFL* miały odpowiednie wartości, spoczywa na aplikacji.

Jest to technika, która może być używana do restartowania grupy komunikatów lub komunikatu logicznego w środku, po wystąpieniu awarii systemu. Po zrestartowaniu systemu aplikacja może ustawić odpowiednie wartości w polach *MDGID*, *MDSEQ*, *MDOFF*, *MDMFL*i *MDPER* , a następnie wywołać komendę MQPUT z parametrem PMSYP lub PMNSYP ustawionym jako *niezbędne*, ale bez określania parametru PMLOGO. Jeśli wywołanie powiedzie się, menedżer kolejek zachowuje informacje o grupie i segmencie, a kolejne wywołania MQPUT używające tego uchwytu kolejki mogą określać PMLOGO jako normalne.

Informacje o grupie i segmencie, które menedżer kolejek przechowuje dla wywołania MQPUT, są oddzielone od informacji o grupie i segmencie, które przechowuje dla wywołania MQGET.

Dla dowolnego uchwytu kolejki aplikacja może swobodnie łączyć wywołania MQPUT, które określają PMLOGO z wywołaniami MQPUT, które nie są, ale należy zauważyć następujące punkty:

- Jeśli parametr PMLOGO nie zostanie określony, każde pomyślne wywołanie MQPUT spowoduje, że menedżer kolejek ustawi informacje o grupie i segmencie dla uchwytu kolejki na wartości określone przez aplikację; spowoduje to zastąpienie istniejących informacji o grupie i segmencie zachowanych przez menedżer kolejek dla uchwytu kolejki.
- Jeśli parametr PMLOGO nie zostanie określony, wywołanie nie zakończy się niepowodzeniem, jeśli istnieje bieżąca grupa komunikatów lub komunikat logiczny. Wywołanie może jednak zakończyć się powodzeniem z kodem zakończenia CCWARN. Tabela 717 na stronie 1214 przedstawia różne obserwacje, które mogą wystąpić. W takich przypadkach, jeśli kod zakończenia nie jest kodem CCOK, kod przyczyny ma jedną z następujących wartości (w zależności od przypadku):
  - RC2241
  - RC2242
  - RC2185
  - RC2245

  Uwaga: Menedżer kolejek nie sprawdza informacji o grupie i segmencie dla wywołania MQPUT1 .

Tabela 717. Wynik, gdy wywołanie MQPUT lub MQCLOSE nie jest spójne z informacjami o grupie i segmencie

Bieżące połączenie to	Poprzednie wywołanie to MQPUT z PMLOGO	Poprzednie wywołanie to MQPUT bez PMLOGO
MQPUT z PMLOGO	CCFAIL (poczta elektroniczna)	CCFAIL (poczta elektroniczna)
MQPUT bez PMLOGO	CWARN	CKOK
MQCLOSE z niezakończoną grupą lub komunikatem logicznym	CWARN	CKOK

Aplikacje, które po prostu chcą umieścić komunikaty i segmenty w porządku logicznym, są zalecane do określenia PMLOGO, ponieważ jest to najprostsza opcja do użycia. Ta opcja zwalnia z konieczności zarządzania informacjami o grupach i segmentach, ponieważ menedżer kolejek zarządza tymi informacjami. Jednak wyspecjalizowane aplikacje mogą wymagać większej kontroli niż ta, którą zapewnia opcja PMLOGO, i można to osiągnąć, nie podając tej opcji. W takim przypadku aplikacja musi upewnić się, że pola *MDGID*, *MDSEQ*, *MDOFF*i *MDMFL* w strukturze MQMD są ustawione poprawnie przed każdym wywołaniem MQPUT lub MQPUT1 .

Na przykład aplikacja, która chce przekazywać odbierane komunikaty fizyczne, bez względu na to, czy są one w grupach, czy w segmentach komunikatów logicznych, nie może określać PMLOGO. Istnieją dwa powody takiej sytuacji:

- Jeśli komunikaty są wczytywane i porządkowane, określenie PMLOGO powoduje przypisanie do komunikatów nowego identyfikatora grupy, co może utrudnić lub uniemożliwić autorowi komunikatów skorelowanie jakichkolwiek komunikatów odpowiedzi lub raportów, które wynikają z grupy komunikatów.
- W złożonej sieci z wieloma ścieżkami między wysyłającymi i odbierającymi menedżerami kolejek komunikaty fizyczne mogą pojawiać się w nieodpowiedniej kolejności. Jeśli w wywołaniu MQGET nie zostanie określona wartość PMLOGO i odpowiadająca jej wartość GMLOGO, aplikacja przekazująca może pobrać i przekazać każdy komunikat fizyczny natychmiast po nadejściu komunikatu, bez konieczności oczekiwania na następny komunikat w porządku logicznym.

Aplikacje generujące komunikaty raportów dla komunikatów w grupach lub segmentach komunikatów logicznych nie mogą również określać PMLOGO podczas umieszczania komunikatu raportu.

PMLOGO można określić z dowolną inną opcją PM*.

Opcje kontekstu: Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

MNOC
Z komunikatem nie jest powiązany żaden kontekst.

Zarówno tożsamość, jak i kontekst źródłowy są ustawione tak, aby wskazywały brak kontekstu. Oznacza to, że pola kontekstu w strukturze MQMD są ustawione na:

- Odstępy dla pól znakowych
- Wartości puste dla pól bajtowych
- Zera dla pól liczbowych

Kod PMDEFC
Użyj domyślnego kontekstu.

Komunikat ma mieć powiązane domyślne informacje o kontekście, zarówno dla tożsamości, jak i dla źródła. Menedżer kolejek ustawia pola kontekstu w deskryptorze komunikatu w następujący sposób:

Tabela 718. Domyślne wartości informacji o kontekście dla pól MQMD

Pole w strukturze MQMD	Użyta wartość
MDUID	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie ustawiana jest wartość pusta.
MDACC	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie ustawiana jest wartość ACNONE.
MDAID	Ustaw na wartość pustą.
MDPAT	Określana na podstawie środowiska.
MDPAN	Określana na podstawie środowiska, jeśli jest to możliwe; w przeciwnym razie ustawiana jest wartość pusta.
MDPD	Ustaw datę umieszczenia komunikatu.

Tabela 718. Domyślne wartości informacji o kontekście dla pól MQMD (kontynuacja)

Pole w strukturze MQMD	Użyta wartość
MDPT	Ustaw czas umieszczenia komunikatu.
MDAOD	Ustaw na wartość pustą.

Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Jest to działanie domyślne, jeśli nie określono opcji kontekstu.

PMPASI
Przekaż kontekst tożsamości z uchwytu kolejki wejściowej.

Komunikat ma mieć powiązane informacje o kontekście. Kontekst tożsamości jest pobierany z uchwytu kolejki określonego w polu *PMCT* . Informacje o kontekście źródłowym są generowane przez menedżer kolejek w taki sam sposób, jak w przypadku PMDEFC (wartości znajdują się w poprzedniej tabeli). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

W przypadku wywołania MQPUT kolejka musi być otwarta za pomocą opcji OOPASI (lub opcji, która ją implikuje). W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją OOPASI.

PMPASA
Przekaż cały kontekst z uchwytu kolejki wejściowej.

Komunikat ma mieć powiązane informacje o kontekście. Zarówno tożsamość, jak i kontekst źródłowy są pobierane z uchwytu kolejki określonego w polu *PMCT* . Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

W przypadku wywołania MQPUT kolejka musi być otwarta za pomocą opcji OOPASA (lub opcji, która ją implikuje). W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją OOPASA.

PSETI
Ustaw kontekst tożsamości z aplikacji.

Komunikat ma mieć powiązane informacje o kontekście. Aplikacja określa kontekst tożsamości w strukturze MQMD. Informacje o kontekście źródłowym są generowane przez menedżer kolejek w taki sam sposób, jak w przypadku PMDEFC (wartości znajdują się w poprzedniej tabeli). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

W przypadku wywołania MQPUT kolejka musi być otwarta za pomocą opcji OOSETI (lub opcji, która ją implikuje). W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją OOSETI.

PSETA
Ustaw cały kontekst z aplikacji.

Komunikat ma mieć powiązane informacje o kontekście. Aplikacja określa tożsamość i kontekst źródłowy w strukturze MQMD. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

W przypadku wywołania MQPUT kolejka musi być otwarta z opcją OOSETA. W przypadku wywołania MQPUT1 wykonywane jest to samo sprawdzenie autoryzacji, co w przypadku wywołania MQOPEN z opcją OOSETA.

Można podać tylko jedną z opcji kontekstu PM*. Jeśli żadna z tych opcji nie zostanie podana, zostanie przyjęta opcja PMDEFC.

Typy odpowiedzi umieszczania. Następujące opcje sterują odpowiedzią zwracaną do wywołania MQPUT lub MQPUT1 . Można podać tylko jedną z tych opcji. Jeśli nie określono PMARES i PMSRES, przyjmuje się PMRASQ lub PMRAST.

PŁETWY

Opcja PMARES żąda wykonania operacji MQPUT lub MQPUT1 bez oczekiwania aplikacji na zakończenie wywołania przez menedżer kolejek. Użycie tej opcji może zwiększyć wydajność przesyłania komunikatów, szczególnie w przypadku aplikacji korzystających z powiązań klienta. Aplikacja może okresowo sprawdzać za pomocą komendy MQSTAT, czy podczas poprzednich wywołań asynchronicznych wystąpił błąd.

W przypadku tej opcji w strukturze MQMD muszą być wypełnione tylko następujące pola:

- MDAID
- MDPAT
- MDPAN
- MDAD

Ponadto, jeśli jako opcje podano jeden lub oba identyfikatory PMNMID lub PMNCID, zwracane identyfikatory MDMID i MDCID również są zakończone. (PMNMID można określić niejawnie, podając puste pole MDMID).

Zostaną wypełnione tylko poprzednio określone pola. Inne informacje, które normalnie byłyby zwracane w strukturze MQMD lub MQPMO, są niezdefiniowane.

W przypadku żądania asynchronicznej odpowiedzi umieszczania dla MQPUT lub MQPUT1opcja CMPCOD i REASON dla CCOK i RCNONE nie musi oznaczać, że komunikat został pomyślnie umieszczony w kolejce. Podczas tworzenia aplikacji MQI korzystającej z asynchronicznej odpowiedzi put i wymagającej potwierdzenia, że komunikaty zostały umieszczone w kolejce, należy sprawdzić zarówno kod CMPCOD, jak i kod PRZYCZYNY operacji put, a także użyć komendy MQSTAT do wysłania zapytania o asynchroniczne informacje o błędzie.

Chociaż powodzenie lub niepowodzenie każdego pojedynczego wywołania MQPUT/MQPUT1 może nie zostać zwrócone natychmiast, pierwszy błąd, który wystąpił w wywołaniu asynchronicznym, może zostać określony w późniejszym momencie za pomocą wywołania MQSTAT.

Jeśli dostarczenie komunikatu trwałego w punkcie synchronizacji nie powiedzie się przy użyciu asynchronicznej odpowiedzi put, a użytkownik podejmie próbę zatwierdzenia transakcji, zatwierdzenie nie powiedzie się, a transakcja zostanie wycofana z kodem zakończenia CCFAIL i przyczyną jest RC2003 . Aplikacja może wywołać MQSTAT w celu określenia przyczyny poprzedniego niepowodzenia MQPUT lub MQPUT1 .

PMSRES

Określenie tej wartości dla opcji put w strukturze MQPMO zapewnia, że operacja MQPUT lub MQPUT1 jest zawsze wykonywana synchronicznie. Jeśli operacja powiedzie się, wszystkie pola w MQMD i MQPMO zostaną zakończone. Jest on udostępniany w celu zapewnienia odpowiedzi synchronicznej niezależnie od domyślnej wartości odpowiedzi umieszczania zdefiniowanej w kolejce lub obiekcie tematu.

MRASQ

Jeśli ta wartość jest określona dla wywołania MQPUT, używany typ odpowiedzi umieszczania jest pobierany z wartości DEFPRESP określonej w kolejce podczas otwierania przez aplikację. Jeśli aplikacja kliencka jest połączona z menedżerem kolejek na poziomie wcześniejszym niż IBM WebSphere MQ 7.0, zachowuje się tak, jakby określono parametr PMSRES.

Jeśli ta opcja jest określona dla wywołania MQPUT1 , wartość DEFPRESP z definicji kolejki nie jest używana. Jeśli wywołanie MQPUT1 używa protokołu PMSYP, zachowuje się tak samo, jak w przypadku protokołu PMARES, a jeśli używa protokołu PMNSYP, zachowuje się tak, jak w przypadku protokołu PMSRES.

PMRAST

Jest to synonim PMRASQ do użycia z obiektami tematów.

Inne opcje: Następujące opcje sterują sprawdzaniem autoryzacji i tym, co się dzieje, gdy menedżer kolejek jest wyciszany:

PMALTU
Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

Wskazuje to, że pole *ODAU* w parametrze **OBJDSC** wywołania MQPUT1 zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności uprawnień do umieszczania komunikatów w kolejce. Wywołanie może zakończyć się pomyślnie tylko wtedy, gdy ta *ODAU* ma uprawnienia do otwarcia kolejki z podanymi opcjami, niezależnie od tego, czy identyfikator użytkownika, pod którym działa aplikacja, jest do tego uprawniony. (Nie ma to jednak zastosowania do określonych opcji kontekstu, które są zawsze sprawdzane względem identyfikatora użytkownika, pod którym działa aplikacja).

Ta opcja jest poprawna tylko w połączeniu z wywołaniem MQPUT1 .

PFIQ
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

Ta opcja wymusza niepowodzenie wywołania MQPUT lub MQPUT1 , jeśli menedżer kolejek jest w stanie wyciszania.

Wywołanie zwraca kod zakończenia CCFAIL z kodem przyczyny RC2161 .

Opcja domyślna: Jeśli żadna z opisanych wcześniej opcji nie jest wymagana, można użyć następującej opcji:

BRAK KZ
Nie określono żadnych opcji.

Ta wartość może być używana do wskazania, że nie określono żadnych innych opcji; wszystkie opcje przyjmują wartości domyślne. Parametr PMNONE jest definiowany w celu wspomagania dokumentacji programu; nie jest planowane, aby ta opcja była używana z innymi opcjami, ale ponieważ jej wartość wynosi zero, takiego użycia nie można wykryć.

Jest to pole wejściowe. Wartością początkową pola *PMOPT* jest PMNONE.

PMPRF (10-cyfrowa liczba całkowita ze znakiem)

Flagi wskazujące, które pola MQPMR są obecne.

To pole zawiera flagi, które muszą być ustawione w celu wskazania, które pola MQPMR są obecne w rekordach umieszczania komunikatów udostępnianych przez aplikację. Parametr *PMPRF* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *PMREC* ma wartość zero lub oba parametry *PMPRO* i *PMPRP* mają wartość zero.

W przypadku pól, które są obecne, menedżer kolejek używa dla każdego miejsca docelowego wartości z pól w odpowiednim rekordzie komunikatu umieszczania. W przypadku pól, które nie są dostępne, menedżer kolejek używa wartości ze struktury MQMD.

Można podać co najmniej jedną z następujących opcji, aby wskazać, które pola są obecne w rekordach komunikatów umieszczania:

Identyfikator PFMID
Pole identyfikatora komunikatu jest obecne.

Identyfikator PFCID
Pole identyfikatora korelacji jest obecne.

Identyfikator PFGID
Pole identyfikatora grupy jest obecne.

PFFB
Pole informacji zwrotnej jest obecne.

FACC
Rozliczanie-pole tokenu jest obecne.

Jeśli ta opcja jest określona, w polu *PMOPT* należy podać wartość PMSETI lub PMSETA. Jeśli ten warunek nie jest spełniony, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2158 .

Jeśli nie ma pól MQPMR, można podać następujące informacje:

PPFNONE
Brak pól rekordu komunikatu umieszczania (put-message).

Jeśli ta wartość jest określona, parametr *PMREC* musi mieć wartość zero lub oba parametry *PMPRO* i *PMPRP* muszą mieć wartość zero.

PFNONE jest zdefiniowane w celu wspomagania dokumentacji programu. Ta stała nie jest przeznaczona do użycia z żadną inną, ale ponieważ jej wartość wynosi zero, nie można wykryć takiego użycia.

Jeśli parametr *PMPRF* zawiera flagi, które nie są poprawne, lub jeśli podano rekordy komunikatu umieszczania, ale parametr *PMPRF* ma wartość PFNONE, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2158 .

Jest to pole wejściowe. Wartością początkową tego pola jest PFNONE. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMPRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie rekordu pierwszego umieszczenia komunikatu od początku MQPMO.

Jest to przesunięcie w bajtach pierwszego rekordu komunikatu umieszczonego przez MQPMR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. Parametr *PMPRO* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *PMREC* ma wartość zero.

Gdy komunikat jest umieszczany na liście dystrybucyjnej, można podać tablicę zawierającą jeden lub więcej rekordów komunikatów MQPMR typu put w celu określenia pewnych właściwości komunikatu dla każdego miejsca docelowego osobno. Są to następujące właściwości:

- Identyfikator komunikatu
- identyfikator korelacji
- Identyfikator grupy
- wartość opinii
- Token rozliczania

Nie jest konieczne określanie wszystkich tych właściwości, ale niezależnie od wybranego podzbioru, pola muszą być podane w poprawnej kolejności. Więcej informacji zawiera opis struktury MQPMR.

Zwykle podczas otwierania listy dystrybucyjnej powinna istnieć taka sama liczba rekordów komunikatów umieszczania, jak liczba rekordów obiektów określonych przez program MQOD. Każdy rekord komunikatów umieszczania dostarcza właściwości komunikatu dla kolejki identyfikowanej przez odpowiedni rekord obiektu. Kolejki na liście dystrybucyjnej, których otwarcie nie powiodło się, muszą mieć przydzielone rekordy komunikatów na odpowiednich pozycjach w tablicy, chociaż w tym przypadku właściwości komunikatu są ignorowane.

Możliwe, że liczba rekordów komunikatów wstawianych różni się od liczby rekordów obiektów. Jeśli istnieje mniej rekordów komunikatów umieszczania niż rekordów obiektów, właściwości komunikatów dla miejsc docelowych, które nie mają rekordów komunikatów umieszczania, są pobierane z odpowiednich pól w deskryptorze MQMD komunikatu. Jeśli istnieje więcej rekordów komunikatów umieszczania niż rekordów obiektów, nadmiar nie jest używany (chociaż dostęp do nich musi być nadal możliwy). Rekordy umieszczania komunikatów są opcjonalne, ale jeśli zostaną podane, musi być ich *PMREC* .

Rekordy komunikatów umieszczania mogą być udostępniane w podobny sposób, jak rekordy obiektów w programie MQOD, przez określenie przesunięcia w programie *PMPRO*lub przez określenie adresu w programie *PMPRP* . Szczegółowe informacje na ten temat zawiera opis pola *ODORO* w sekcji “MQOD (deskryptor obiektu) w systemie IBM i” na stronie 1193.

Można użyć nie więcej niż jednej z wartości *PMPRO* i *PMPRP* ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2159 , jeśli obie wartości są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMPRP (wskaźnik)

Adres pierwszego rekordu umieszczonego komunikatu.

Jest to adres pierwszego rekordu umieszczenia komunikatu MQPMR. Parametr *PMPRP* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *PMREC* ma wartość zero.

Do określenia rekordów umieszczania komunikatów można użyć wartości *PMPRP* lub *PMPRO* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola PMRRO . Jeśli parametr *PMPRP* nie jest używany, musi być ustawiony na wskaźnik pusty lub bajty puste.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMREC (10-cyfrowa liczba całkowita ze znakiem)

Liczba rekordów komunikatów umieszczania lub rekordów odpowiedzi.

Jest to liczba rekordów komunikatów umieszczenia MQPMR lub rekordów odpowiedzi MQRR, które zostały udostępnione przez aplikację. Ta liczba może być większa od zera tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Rekordy komunikatów umieszczania i rekordy odpowiedzi są opcjonalne-aplikacja nie musi udostępniać żadnych rekordów lub może udostępnić rekordy tylko jednego typu. Jeśli jednak aplikacja udostępnia rekordy obu typów, musi udostępniać rekordy *PMREC* każdego typu.

Wartość *PMREC* nie musi być taka sama, jak liczba miejsc docelowych na liście dystrybucyjnej. Jeśli podano zbyt wiele rekordów, nadmiarowe rekordy nie są używane. Jeśli podano zbyt mało rekordów, używane są wartości domyślne dla właściwości komunikatów dla miejsc docelowych, które nie zawierają rekordów komunikatów (patrz sekcja *PMPRO* w dalszej części tego tematu).

Jeśli wartość *PMREC* jest mniejsza od zera lub większa od zera, ale komunikat nie jest umieszczany na liście dystrybucyjnej, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2154 .

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMRMN (48-bajtowy łańcuch znaków)

Przetłumaczona nazwa docelowego menedżera kolejek.

Jest to nazwa docelowego menedżera kolejek po wykonaniu tłumaczenia nazw przez lokalny menedżer kolejek. Zwracana nazwa jest nazwą menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez produkt *PMRQN*, i może być nazwą lokalnego menedżera kolejek.

Jeśli *PMRQN* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejek, do której należy lokalny menedżer kolejek, *PMRMN* jest nazwą grupy współużytkowania kolejek. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, parametr *PMRQN* może być nazwą grupy współużytkowania kolejek lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejek (rodzaj zwracanej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

Jest to pole wyjściowe. Długość tego pola jest określona przez LNQMN. Wartością początkową tego pola jest 48 znaków odstępu.

PMRQN (48-bajtowy łańcuch znaków)

Przetłumaczona nazwa kolejki docelowej.

Jest to nazwa kolejki docelowej po wykonaniu tłumaczenia nazw przez menedżera kolejek lokalnych. Zwracana nazwa jest nazwą kolejki, która istnieje w menedżerze kolejek identyfikowanym przez *PMRMN*.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką; jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

Jest to pole wyjściowe. Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

PMRRO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie pierwszego rekordu odpowiedzi od początku MQPMO.

Jest to przesunięcie w bajtach pierwszego rekordu odpowiedzi MQRR od początku struktury MQPMO. Przesunięcie może być dodatnie lub ujemne. Parametr *PMRRO* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *PMREC* ma wartość zero.

Gdy komunikat jest umieszczany na liście dystrybucyjnej, można podać tablicę co najmniej jednego rekordu odpowiedzi MQRR w celu zidentyfikowania kolejek, do których komunikat nie został pomyślnie wysłany (pole*RRCC* w MQRR), oraz przyczyny każdego niepowodzenia (pole*RRREA* w MQRR). Komunikat mógł nie zostać wysłany, ponieważ otwarcie kolejki nie powiodło się lub operacja umieszczania nie powiodła się. Menedżer kolejek ustawia rekordy odpowiedzi tylko wtedy, gdy wynik wywołania jest mieszany (to znaczy niektóre komunikaty zostały wysłane pomyślnie, podczas gdy inne nie powiodły się lub wszystkie nie powiodły się, ale z różnych przyczyn). Kod przyczyny RC2136 z wywołania wskazuje ten przypadek. Jeśli ten sam kod przyczyny ma zastosowanie do wszystkich kolejek, przyczyna jest zwracana w parametrze **REASON** wywołania MQPUT lub MQPUT1 , a rekordy odpowiedzi nie są ustawiane.

Zwykle liczba rekordów odpowiedzi powinna być taka sama, jak liczba rekordów obiektów określonych przez MQOD podczas otwierania listy dystrybucyjnej. Jeśli jest to konieczne, każdy rekord odpowiedzi jest ustawiany na kod zakończenia i kod przyczyny dla operacji umieszczania w kolejce identyfikowanej przez odpowiedni rekord obiektu. Kolejki na liście dystrybucyjnej, których otwarcie nie powiodło się, muszą nadal mieć przydzielone rekordy odpowiedzi na odpowiednich pozycjach w tablicy, chociaż są one ustawione na kod zakończenia i kod przyczyny wynikający z operacji otwarcia, a nie operacji umieszczania (put).

Liczba rekordów odpowiedzi może różnić się od liczby rekordów obiektów. Jeśli liczba rekordów odpowiedzi jest mniejsza niż liczba rekordów obiektów, aplikacja może nie być w stanie zidentyfikować wszystkich miejsc docelowych, dla których operacja umieszczania nie powiodła się, lub przyczyn niepowodzeń. Jeśli liczba rekordów odpowiedzi jest większa niż liczba rekordów obiektów, nadwyżka nie jest używana (chociaż dostęp do niej musi być nadal możliwy). Rekordy odpowiedzi są opcjonalne, ale jeśli zostaną podane, musi być ich *PMREC* .

Rekordy odpowiedzi mogą być udostępniane w podobny sposób, jak rekordy obiektów w MQOD, przez określenie przesunięcia w *PMRRO*lub przez określenie adresu w *PMRRP* ; Szczegółowe informacje na ten temat zawiera opis pola *ODORO* w sekcji “MQOD (deskryptor obiektu) w systemie IBM i” na stronie 1193. Można jednak użyć nie więcej niż jednej z wartości *PMRRO* i *PMRRP* ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2156 , jeśli obie wartości są niezerowe.

W przypadku wywołania MQPUT1 to pole musi mieć wartość zero. Dzieje się tak, ponieważ informacje o odpowiedzi (jeśli są żądane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

Jest to pole wejściowe. Wartością początkową tego pola jest 0. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMRRP (wskaźnik)

Adres pierwszego rekordu odpowiedzi.

Jest to adres pierwszego rekordu odpowiedzi MQRR. Parametr *PMRRP* jest używany tylko wtedy, gdy komunikat jest umieszczany na liście dystrybucyjnej. Pole jest ignorowane, jeśli parametr *PMREC* ma wartość zero.

Do określenia rekordów odpowiedzi można użyć wartości *PMRRP* lub *PMRRO* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola PMRRO . Jeśli parametr *PMRRP* nie jest używany, musi być ustawiony na wskaźnik pusty lub bajty puste.

W przypadku wywołania MQPUT1 to pole musi być wskaźnikiem pustym lub bajtami o wartości NULL. Dzieje się tak, ponieważ informacje o odpowiedzi (jeśli są żądane) są zwracane w rekordach odpowiedzi określonych przez deskryptor obiektu MQOD.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty. To pole jest ignorowane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

PMSIDV
Identyfikator struktury opcji put-message.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest PMSIDV.

PMSL (MQLONG)

Poziom subskrypcji, której dotyczy ta publikacja.

Tę publikację otrzymują tylko te subskrypcje, których najwyższa wartość atrybutu *PMSL* jest mniejsza lub równa tej wartości. Wartość ta musi mieścić się w zakresie od 0 do 9; zero jest najniższym poziomem.

Wartością początkową tego pola jest 9.

PMTO (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone; jego wartość nie jest istotna. Wartością początkową tego pola jest -1.

PMUDC (10-cyfrowa liczba całkowita ze znakiem)

Liczba komunikatów pomyślnie wysłanych do kolejek zdalnych.

Jest to liczba komunikatów pomyślnie wysłanych przez bieżące wywołanie MQPUT lub MQPUT1 do kolejek znajdujących się na liście dystrybucyjnej, które są tłumaczone na kolejki zdalne. Komunikaty przechowywane tymczasowo przez menedżera kolejek w postaci listy dystrybucyjnej są liczone jako liczba pojedynczych miejsc docelowych, które zawierają te listy dystrybucyjne. To pole jest również ustawiane podczas umieszczania komunikatu w pojedynczej kolejce, która nie znajduje się na liście dystrybucyjnej.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0. To pole nie jest ustawiane, jeśli wartość *PMVER* jest mniejsza niż PMVER2.

PMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być jedną z następujących wartości:

PMVER1
Version-1 struktura opcji put-message.

PMVER2
Version-2 struktura opcji put-message.

Pola, które istnieją tylko w najnowszej wersji struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

PMVERC
Bieżąca wersja struktury opcji put-message.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest PMVER1.

Wartości początkowe

Tabela 719. Pola w MQPMO

Nazwa pola	Nazwa stałej	Wartość stałej
PMSID	PMSIDV	'PMO¬'
PMVER	PMVER1	1
PMOPT	BRAK KZ	0
PMTO	Brak	-1
PMCT	Brak	0
PMKDC	Brak	0
PMUDC	Brak	0
PMIDC	Brak	0
PMRQN	Brak	Puste
PMRMN	Brak	Puste
PMREC	Brak	0
PMPRF	PPFNONE	0
PMPRO	Brak	0
PMRRO	Brak	0
PMPRP	Brak	Pusty wskaźnik lub puste bajty
PMRRP	Brak	Pusty wskaźnik lub puste bajty

Uwaga:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQPMO Structure
D*
D* Structure identifier
D  PMSID                  1      4    INZ('PMO ')
D* Structure version number
D  PMVER                  5      8I 0 INZ(1)
D* Options that control the action of MQPUT and MQPUT1
D  PMOPT                  9     12I 0 INZ(0)
D* Reserved
D  PMTO                  13     16I 0 INZ(-1)
D* Object handle of input queue
D  PMCT                  17     20I 0 INZ(0)
D* Number of messages sent successfully to local queues
D  PMKDC                 21     24I 0 INZ(0)
D* Number of messages sent successfully to remote queues
D  PMUDC                 25     28I 0 INZ(0)
D* Number of messages that could notbe sent
D  PMIDC                 29     32I 0 INZ(0)
D* Resolved name of destination queue
D  PMRQN                 33     80    INZ
D* Resolved name of destination queue manager
D  PMRMN                 81    128    INZ
D* Number of put message records or response records present
D  PMREC                129    132I 0 INZ(0)
D* Flags indicating which MQPMR fields are present
D  PMPRF                133    136I 0 INZ(0)
D* Offset of first put message record from start of MQPMO
```

```
     D  PMPRO                137    140I 0 INZ(0)
     D* Offset of first response record from start of MQPMO
     D  PMRRO                141    144I 0 INZ(0)
     D* Address of first put message record
     D  PMPRP                145    160*   INZ(*NULL)
     D* Address of first response record
     D  PMRRP                161    176*   INZ(*NULL)
     D* Original message handle
     D  PMOMH                177    184I 0
     D* New message handle
     D  PMNMH                185    190I 0
     D* The action being performed
     D  PMACT                191    194I 0
     D* Reserved
     D  PMRE1                195    198I 0
```

IBM i MQPMR (Put-message record) w systemie IBM i

Struktura MQPMR służy do określania różnych właściwości komunikatu dla pojedynczego miejsca docelowego, gdy komunikat jest umieszczany na liście dystrybucyjnej.

Przegląd

Cel: MQPMR to struktura wejścia/wyjścia dla wywołań MQPUT i MQPUT1 .

Zestaw znaków i kodowanie: dane w MQPMR muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Użycie: Poprzez udostępnienie tablicy tych struktur w wywołaniu MQPUT lub MQPUT1 można określić inne wartości dla każdej kolejki docelowej na liście dystrybucyjnej. Niektóre pola są tylko polami wejściowymi, inne są polami wejściowymi i wyjściowymi.

Uwaga: Ta struktura jest nietypowa, ponieważ nie ma stałego układu. Pola w tej strukturze są opcjonalne, a obecność lub nieobecność każdego pola jest wskazywana przez flagi w polu *PMPRF* w MQPMO. Pola, które są obecne w produkcie ***, muszą występować w następującej kolejności*** :

- *PRMID*
- *PRCID*
- *PRGID*
- *PRFB*
- *PRACC*

Nieobecne pola nie zajmują miejsca w rekordzie.

Ponieważ MQPMR nie ma stałego układu, w pliku COPY nie jest dostępna jego definicja. Programista aplikacji powinien utworzyć deklarację zawierającą pola, które są wymagane przez aplikację, i ustawić flagi w pliku *PMPRF* , aby wskazać pola, które są obecne.

- "Pola" na stronie 1224
- "Wartości początkowe" na stronie 1226
- "Deklaracja RPG" na stronie 1226

Pola

Struktura MQPMR zawiera następujące pola; są one opisane w **porządku alfabetycznym**:

PRACC (32-bitowy łańcuch bitowy)

Token rozliczania.

Jest to token rozliczania, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *MDACC* w strukturze MQMD dla

operacji umieszczania w pojedynczej kolejce. Więcej informacji na temat zawartości tego pola zawiera opis *MDACC* w sekcji "MQMD (deskryptor komunikatu) w systemie IBM i" na stronie 1141 .

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

Jest to pole wejściowe.

PRCID (24-bajtowy łańcuch bitowy)

Identyfikator korelacji.

Jest to identyfikator korelacji, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *MDCID* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie występuje w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowych, wartość w MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *PRCID* .

Jeśli określono PMNCID, *pojedynczy* nowy identyfikator korelacji jest generowany i używany dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania PMNMID (patrz pole *PRMID*).

Jest to pole wejściowe/wyjściowe.

PRFB (10-cyfrowa liczba całkowita ze znakiem)

Informacja zwrotna lub kod przyczyny.

Jest to kod informacji zwrotnej, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *MDFB* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie jest obecne, używana jest wartość w strukturze MQMD.

Jest to pole wejściowe.

PRGID (24-bajtowy łańcuch bitowy)

Identyfikator grupy.

Jest to identyfikator grupy, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR podanej w wywołaniu MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *MDGID* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie występuje w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowych, wartość w MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *PRGID* . Wartość jest przetwarzana zgodnie z opisem w sekcji Tabela 716 na stronie 1212, ale z następującymi różnicami:

- W przypadku użycia nowego identyfikatora grupy menedżer kolejek generuje inny identyfikator grupy dla każdego miejsca docelowego (tzn. żadne dwa miejsca docelowe nie mają tego samego identyfikatora grupy).
- W przypadku użycia wartości w polu wywołanie kończy się niepowodzeniem z kodem przyczyny RC2258.

Jest to pole wejściowe/wyjściowe.

PRMID (24-bajtowy łańcuch bitowy)

Identyfikator komunikatu.

Jest to identyfikator komunikatu, który ma być używany dla komunikatu wysyłanego do kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu

MQOPEN lub MQPUT1 . Jest on przetwarzany w taki sam sposób, jak pole *MDMID* w strukturze MQMD dla operacji umieszczania w pojedynczej kolejce.

Jeśli to pole nie występuje w rekordzie MQPMR lub istnieje mniej rekordów MQPMR niż miejsc docelowych, wartość w MQMD jest używana dla tych miejsc docelowych, które nie mają rekordu MQPMR zawierającego pole *PRMID* . Jeśli ta wartość to MINONE, dla *każdego* z tych miejsc docelowych generowany jest nowy identyfikator komunikatu (tzn. żadne dwa z tych miejsc docelowych nie mają takiego samego identyfikatora komunikatu).

Jeśli określono PMNMID, nowe identyfikatory komunikatów są generowane dla wszystkich miejsc docelowych na liście dystrybucyjnej, niezależnie od tego, czy mają one rekordy MQPMR. Różni się to od sposobu przetwarzania PMNCID (patrz pole *PRCID*).

Jest to pole wejściowe/wyjściowe.

Wartości początkowe

Dla tej struktury nie zdefiniowano wartości początkowych, ponieważ nie podano deklaracji struktury. Poniższa przykładowa deklaracja pokazuje, w jaki sposób struktura powinna być deklarowana przez programistę aplikacji, jeśli wszystkie pola są wymagane.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQPMR Structure
D*
D* Message identifier
D  PRMID                  1     24
D* Correlation identifier
D  PRCID                 25     48
D* Group identifier
D  PRGID                 49     72
D* Feedback or reason code
D  PRFB                  73     76I 0
D* Accounting token
D  PRACC                 77    108
```

IBM i MQRFH (reguły i nagłówek formatowania) w produkcie IBM i

Struktura MQRFH definiuje układ reguł i nagłówka formatowania.

Przegląd

Cel: Ten nagłówek może być używany do wysyłania danych łańcuchowych w postaci par nazwa-wartość.

Nazwa formatu: FMRFH.

Zestaw znaków i kodowanie: pola w strukturze MQRFH (w tym *RFNVS*) są w zestawie znaków i kodowaniu określonym przez pola *MDCSI* i *MDENC* w strukturze nagłówka poprzedzającej MQRFH lub przez te pola w strukturze MQMD, jeśli MQRFH jest na początku danych komunikatu aplikacji.

Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

- “Pola” na stronie 1226
- “Wartości początkowe” na stronie 1229
- “Deklaracja RPG” na stronie 1229

Pola

Struktura MQRFH zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

RFCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych następujących po *RFNVS*.

Określa identyfikator zestawu znaków danych następujących po *RFNVS* ; nie ma zastosowania do danych znakowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola *MDPAT* w MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

Kodowanie liczbowe danych następujących po *RFNVS*.

Określa kodowanie liczbowe danych następujących po *RFNVS* ; nie ma zastosowania do danych liczbowych w samej strukturze MQRFH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest ENNAT.

RFFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Można określić następujące elementy:

BRAK RFNONE
Brak flag.

Wartością początkową tego pola jest RFNONE.

RFFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych następująca po *RFNVS*.

Określa nazwę formatu danych następujących po *RFNVS*.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

RFLEN (10-cyfrowa liczba całkowita ze znakiem)

Łączna długość MQRFH z uwzględnieniem *RFNVS*.

Jest to długość (w bajtach) struktury MQRFH, łącznie z polem *RFNVS* na końcu struktury. Długość nie obejmuje żadnych danych użytkownika, które następują po polu *RFNVS* .

Aby uniknąć problemów z konwersją danych użytkownika w niektórych środowiskach, należy rozważyć użycie *RFLEN* jako wielokrotności czterech.

Następująca stała określa długość *stałej* części struktury, czyli długość bez pola *RFNVS* :

RFLENV (flaga)
Długość stałej części struktury MQRFH.

Wartością początkową tego pola jest RFLENV.

RFNVS (n-bajtowy łańcuch znaków)

Łańcuch zawierający pary nazwa-wartość.

Jest to łańcuch znaków o zmiennej długości, zawierający pary nazwa-wartość w postaci:

```
name1 value1 name2 value2 name3 value3 ...
```

Każda nazwa lub wartość musi być oddzielona od sąsiedniej nazwy lub wartości jednym lub większą liczbą znaków odstępu. Te znaki odstępu nie są istotne. Nazwa lub wartość może zawierać znaczące odstępy, poprzedzone i opatrzone przyrostkiem nazwy lub wartości znakiem cudzysłowu. Wszystkie znaki między otwierającym i zamykającym znakiem cudzysłowu są traktowane jako znaczące.
W poniższym przykładzie nazwa to FAMOUS_WORDS, a wartość to `Hello World`:

```
FAMOUS_WORDS "Hello World"
```

Nazwa lub wartość może zawierać dowolne znaki inne niż znak o kodzie zero (który działa jako separator w przypadku *RFNVS*). Jednak w celu ułatwienia współdziałania aplikacja może preferować ograniczenie nazw do następujących znaków:

- Pierwszy znak: wielkie lub małe litery (od A do Z lub od a do z) lub podkreślenie.
- Kolejne znaki: wielka lub mała litera, cyfra dziesiętna (od 0 do 9), podkreślenie, łącznik lub kropka.

Jeśli nazwa lub wartość zawiera jeden lub więcej znaków cudzysłowu, nazwa lub wartość musi być ujęta w cudzysłów, a każdy znak cudzysłowu w łańcuchu musi być podwojony:

```
Famous_Words "The program displayed ""Hello World"""
```

W nazwach i wartościach rozróżniana jest wielkość liter, tzn. małe litery nie są traktowane jako wielkie litery. Na przykład FAMOUS_WORDS i `Famous_Words` to dwie różne nazwy.

Długość w bajtach *RFNVS* jest równa *RFLEN* minus RFLENV. Aby uniknąć problemów z konwersją danych użytkownika w niektórych środowiskach, zaleca się, aby ta długość była wielokrotnością czterech. Łańcuch *RFNVS* musi być dopełniony spacjami do tej długości lub zakończony wcześniej przez umieszczenie znaku o kodzie zero po ostatnim istotnym znaku w łańcuchu. Znak o kodzie zero i następujące po nim bajty (do określonej długości *RFNVS*) są ignorowane.

Uwaga: Ponieważ długość tego pola nie jest stała, pole jest pomijane w deklaracjach struktury, które są udostępniane dla obsługiwanych języków programowania.

RFSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

RFSIDV,
Identyfikator struktury nagłówka reguł i formatowania.

Wartością początkową tego pola jest RFSIDV.

RFVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

RFVER1
Version-1 -reguły i struktura nagłówka formatowania.

Wartością początkową tego pola jest RFVER1.

Wartości początkowe

Tabela 720. Pola w MQRFH		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
RFSID	RFSIDV,	'RFH¬'
RFVER	RFVER1	1
RFLEN	RFLENV (flaga)	32
RFENC	ENNAT,	Zależy od środowiska
RFCSI	CSUNDF	0
RFFMT	BRAK FMNONE	Puste
RFFLG	BRAK RFNONE	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQRFH Structure
D*
D* Structure identifier
D  RFSID                  1      4    INZ('RFH ')
D* Structure version number
D  RFVER                  5      8I 0 INZ(1)
D* Total length of MQRFH includingNameValueString
D  RFLEN                  9     12I 0 INZ(32)
D* Numeric encoding of data that followsNameValueString
D  RFENC                 13     16I 0 INZ(273)
D* Character set identifier of data thatfollows NameValueString
D  RFCSI                 17     20I 0 INZ(0)
D* Format name of data that followsNameValueString
D  RFFMT                 21     28    INZ('        ')
D* Flags
D  RFFLG                 29     32I 0 INZ(0)
```

IBM i MQRFH2 (reguły i nagłówek formatowania 2) w systemie IBM i

Struktura MQRFH2 definiuje format reguł i nagłówka formatowania version-2 .

Przegląd

Cel: Ten nagłówek może być używany do wysyłania danych, które zostały zakodowane przy użyciu składni podobnej do XML. Komunikat może zawierać co najmniej dwie struktury MQRFH2 w serii, z danymi użytkownika opcjonalnie po ostatniej strukturze MQRFH2 w serii.

Nazwa formatu: FMRFH2.

Zestaw znaków i kodowanie: specjalne reguły mają zastosowanie do zestawu znaków i kodowania używanego w strukturze MQRFH2 :

- Pola inne niż *RF2NVD* są w zestawie znaków i kodowaniu, które są nadawane przez pola *MDCSI* i *MDENC* w strukturze nagłówka poprzedzającej nagłówek MQRFH2, lub przez te pola w strukturze MQMD, jeśli na początku danych komunikatu aplikacji znajduje się nagłówek MQRFH2 .

  Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Jeśli w wywołaniu MQGET określono wartość GMCONV, menedżer kolejek przekształca te pola w żądany zestaw znaków i kodowanie.

- *RF2NVD* znajduje się w zestawie znaków podanym w polu *RF2NVC* . Tylko niektóre zestawy znaków Unicode są poprawne dla *RF2NVC* (szczegółowe informacje zawiera opis *RF2NVC*).

  Niektóre zestawy znaków mają reprezentację zależną od kodowania. Jeśli *RF2NVC* jest jednym z tych zestawów znaków, kodowanie *RF2NVD* musi być takie samo jak kodowanie innych pól w MQRFH2.

  Jeśli w wywołaniu MQGET określono wartość GMCONV, menedżer kolejek przekształca wartość *RF2NVD* w żądane kodowanie, ale nie zmienia swojego zestawu znaków.

- "Pola" na stronie 1230
- "Wartości początkowe" na stronie 1235
- "Deklaracja RPG" na stronie 1235

Pola

Struktura MQRFH2 zawiera następujące pola. Pola są opisane w porządku alfabetycznym:

RF2CSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych następujących po ostatnim polu *RF2NVD* .

Określa identyfikator zestawu znaków danych następujących po ostatnim polu *RF2NVD* . Nie ma zastosowania do danych znakowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola *MDPAT* w MQMD jest ATBRKR.

Wartością początkową tego pola jest CSINHT.

RF2ENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe danych następujących po ostatnim polu *RF2NVD* .

Określa kodowanie liczbowe danych następujących po ostatnim polu *RF2NVD* . Nie ma ono zastosowania do danych liczbowych w samej strukturze MQRFH2 .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest ENNAT.

RF2FLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi.

Należy podać następującą wartość:

BRAK RFNONE
Brak flag.

Wartością początkową tego pola jest RFNONE.

RF2FMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych następujących po ostatnim polu *RF2NVD* .

Określa nazwę formatu danych następujących po ostatnim polu *RF2NVD* .

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

RF2LEN (10-cyfrowa liczba całkowita ze znakiem)

Łączna długość nagłówka MQRFH2 z uwzględnieniem wszystkich pól *RF2NVL* i *RF2NVD* .

Jest to długość w bajtach struktury MQRFH2 z uwzględnieniem pól *RF2NVL* i *RF2NVD* na końcu struktury. Poprawne jest, że na końcu struktury znajduje się wiele par pól *RF2NVL* i *RF2NVD* w sekwencji:

```
length1, data1, length2, data2, ...
```

RF2LEN nie zawiera żadnych danych użytkownika, które mogą występować po ostatnim polu *RF2NVD* na końcu struktury.

Aby uniknąć problemów z konwersją danych użytkownika w niektórych środowiskach, należy rozważyć użycie *RF2LEN* jako wielokrotności czterech.

Następująca stała określa długość *stałej* części struktury, czyli długość z wyłączeniem pól *RF2NVL* i *RF2NVD* :

RFLEN2
Długość stałej części struktury MQRFH2 .

Wartością początkową tego pola jest RFLEN2.

RF2NVC (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków *RF2NVD*.

Określa identyfikator kodowanego zestawu znaków dla danych w polu *RF2NVD* . Różni się to od zestawu znaków innych łańcuchów w strukturze MQRFH2 i może różnić się od zestawu znaków danych (jeśli istnieje), który występuje po ostatnim polu *RF2NVD* na końcu struktury.

RF2NVC musi mieć jedną z następujących wartości CCSID:

1200
UTF-16, najnowsza obsługiwana wersja Unicode

13488
UTF-16, podzbiór Unicode w wersji 2.0

17584
Podzbiór UTF-16, wersja Unicode 3.0 (zawiera symbol euro)

1208
UTF-8, najnowsza obsługiwana wersja Unicode

W przypadku zestawów znaków UTF-16 kodowanie (kolejność bajtów) w pliku *RF2NVD* musi być takie samo jak kodowanie innych pól w strukturze MQRFH2 . Znaki zastępcze (X'D800'do X'DFFF') nie są obsługiwane.

Uwaga: Jeśli komenda *RF2NVC* nie ma żadnej z wymienionych wcześniej wartości, a struktura MQRFH2 wymaga konwersji w wywołaniu MQGET, wywołanie zostanie zakończone z kodem przyczyny RC2111 , a komunikat zostanie zwrócony bez konwersji.

Wartością początkową tego pola jest 1208.

RF2NVD (n-bajtowy łańcuch znaków)

Nazwa/wartość danych.

Jest to łańcuch znaków o zmiennej długości, zawierający dane zakodowane przy użyciu składni podobnej do składni XML. Długość tego łańcucha w bajtach jest podana w polu *RF2NVL* , które poprzedza pole *RF2NVD* . Długość ta powinna być wielokrotnością liczby cztery.

Pola *RF2NVL* i *RF2NVD* są opcjonalne, ale jeśli występują, muszą występować jako para i być sąsiadujące. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

```
length1 data1 length2 data2 length3 data3
```

Ponieważ te pola są opcjonalne, są pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.

Wartość *RF2NVD* jest nietypowa, ponieważ nie jest przekształcana w zestaw znaków określony w wywołaniu MQGET, gdy komunikat jest wczytywany z opcją GMCONV; wartość *RF2NVD* pozostaje w oryginalnym zestawie znaków. Jednak wartość *RF2NVD* jest przekształcana w kodowanie określone w wywołaniu MQGET.

Składnia danych nazwa/wartość: Łańcuch składa się z pojedynczego "folderu", który zawiera zero lub więcej właściwości. Folder jest ograniczony znacznikami początkowym i końcowym XML o takiej samej nazwie jak folder:

```
<folder> property1 property2 ... </folder>
```

Znaki następujące po znaczniku końcowym folderu, do długości zdefiniowanej przez *RF2NVL*, muszą być puste. W folderze każda właściwość składa się z nazwy i wartości oraz opcjonalnie z typu danych:

```
<name dt="datatype">value</name>
```

W tych przykładach:

- Znaki separatora (<, =, ",/i >) muszą być podane dokładnie tak, jak pokazano.
- `name` jest nazwą właściwości określoną przez użytkownika. Więcej informacji na temat nazw można znaleźć w poniższym przykładzie.
- `datatype` jest opcjonalnym określonym przez użytkownika typem danych właściwości. Poprawne typy danych można znaleźć w poniższym przykładzie.
- `value` jest wartością właściwości określoną przez użytkownika. Więcej informacji na temat wartości można znaleźć w poniższych akapitach.
- Odstępy są istotne między znakiem > poprzedzającym wartość i znakiem < występującym po wartości, a co najmniej jedna spacja musi poprzedzać znak `dt=`. W innych miejscach odstępy mogą być kodowane dowolnie między znacznikami lub przed lub po znacznikach (na przykład w celu zwiększenia czytelności). Odstępy te nie są istotne.

Jeśli właściwości są ze sobą powiązane, można je pogrupować, umieszczając je w początkowych i końcowych znacznikach XML o takiej samej nazwie jak nazwa grupy:

```
<folder> <group> property1 property2 ... </group> </folder>
```

Grupy mogą być zagnieżdżone w innych grupach bez ograniczeń, a grupa może występować w folderze więcej niż jeden raz. Poprawne jest również, aby folder zawierał niektóre właściwości w grupach, a inne nie w grupach.

Nazwy właściwości, grup i folderów: Nazwy właściwości, grup i folderów muszą być poprawnymi nazwami znaczników XML, z wyjątkiem znaku dwukropka, który nie jest dozwolony w nazwie właściwości, grupy lub folderu. W szczególności:

- Nazwy muszą zaczynać się od litery lub znaku podkreślenia. Poprawne litery są zdefiniowane w specyfikacji XML W3C i składają się zasadniczo z kategorii Unicode Ll, Lu, Lo, Lt i Nl.

- Pozostałe znaki w nazwie mogą być literami, cyframi dziesiętnymi, podkreśleniami, myślnikami lub kropkami. Odpowiadają one kategoriom Unicode Ll, Lu, Lo, Lt, Nl, Mc, Mn, Lm i Nd.
- Znaki zgodności Unicode (X'F900' i nowsze) nie są dozwolone w żadnej części nazwy.
- Nazwy nie mogą rozpoczynać się od łańcucha XML ani zaczynać się od wielkich, ani małych liter.

Ponadto:

- W nazwach rozróżniana jest wielkość liter. Na przykład ABC, abci Abc to trzy różne nazwy.
- Każdy folder ma oddzielną przestrzeń nazw. W wyniku tego grupa lub właściwość w jednym folderze nie powoduje konfliktu z grupą lub właściwością o tej samej nazwie w innym folderze.
- Grupy i właściwości zajmują tę samą przestrzeń nazw w folderze. W wyniku tego właściwość nie może mieć takiej samej nazwy jak grupa w folderze zawierającym tę właściwość.

Ogólnie rzecz biorąc, programy analizujące pole *RF2NVD* powinny ignorować właściwości lub grupy o nazwach, których program nie rozpoznaje, pod warunkiem, że te właściwości lub grupy są poprawnie sformatowane.

Typy danych właściwości: Każda właściwość może mieć opcjonalny typ danych. Jeśli określono, typ danych musi być jedną z następujących wartości (wielkimi, małymi lub mieszanymi literami):

Tabela 721. Typy danych i ich użycie

Typ danych	**Zastosowanie**
`string`	Dowolna sekwencja znaków. Niektóre znaki muszą być określone przy użyciu sekwencji o zmienionym znaczeniu.
`boolean`	Znak 0 lub 1 (1 oznacza TRUE).
`bin.hex`	Cyfry szesnastkowe reprezentujące oktety.
`i1`	Liczba całkowita z zakresu od -128 do +127, wyrażona tylko cyframi dziesiętnymi i opcjonalnym znakiem.
`i2`	Liczba całkowita z zakresu od -32 768 do +32 767, wyrażona tylko cyframi dziesiętnymi i opcjonalnym znakiem.
`i4`	Liczba całkowita z zakresu od -2 147 483 648 do + 2 147 483 647, wyrażona tylko cyframi dziesiętnymi i opcjonalnym znakiem.
`i8`	Liczba całkowita z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, wyrażona tylko cyframi dziesiętnymi i opcjonalnym znakiem.
`int`	Liczba całkowita z zakresu od -9 223 372 036 854 775 808 do + 9 223 372 036 854 775 807, wyrażona tylko cyframi dziesiętnymi i opcjonalnym znakiem. Można go użyć zamiast `i1`, `i2`, `i4`lub `i8` , jeśli nadawca nie chce sugerować konkretnej precyzji.
`r4`	Liczba zmiennopozycyjna o wielkości z zakresu od 1.175E-37 do 3.402 823 47E+38, wyrażona za pomocą cyfr dziesiętnych, opcjonalnego znaku, opcjonalnych cyfr ułamkowych i opcjonalnego wykładnika.

Tabela 721. Typy danych i ich użycie (kontynuacja)	
Typ danych	**Zastosowanie**
r8	Liczba zmiennopozycyjna o wielkości z zakresu od 2.225E-307 do 1.797 693 134 862 3E+308 wyrażona jako cyfry dziesiętne, opcjonalny znak, opcjonalne cyfry ułamkowe i opcjonalny wykładnik.

Wartości właściwości: wartość właściwości może składać się z dowolnych znaków, z wyjątkiem znaków specjalnych, z którymi powiązana jest obowiązkowa sekwencja o zmienionym znaczeniu. Każde wystąpienie w wartości znaku oznaczonego w poniższej tabeli jako "obowiązkowy" musi zostać zastąpione odpowiednią sekwencją o zmienionym znaczeniu. Tabela zawiera również znaki, z którymi powiązana jest opcjonalna sekwencja o zmienionym znaczeniu. Każde wystąpienie w wartości znaku oznaczonego jako "opcjonalny" można zastąpić odpowiednią sekwencją o zmienionym znaczeniu, ale nie jest to wymagane.

Tabela 722. Znaki o zmienionym znaczeniu i ich użycie		
Znak	**Sekwencja o zmienionym znaczeniu**	**Użycie**
&	&amp;	Obowiązkowy
<	<	Obowiązkowy
>	&gt;	Opcjonalne
"	&quot;	Opcjonalne
'	&apos;	Opcjonalne

Uwaga: Znak & na początku sekwencji o zmienionym znaczeniu nie może zostać zastąpiony znakiem &.

W poniższym przykładzie spacje w wartości są istotne, jednak nie są potrzebne żadne sekwencje o zmienionym znaczeniu:

```
<Famous_Words>The program displayed "Hello World"</Famous_Words>
```

RF2NVL (10-cyfrowa liczba całkowita ze znakiem)

Długość *RF2NVD*.

Określa długość (w bajtach) danych w polu *RF2NVD* . Aby uniknąć problemów z konwersją danych (jeśli występuje) po polu *RF2NVD* , wartość *RF2NVL* powinna być wielokrotnością liczby cztery.

Uwaga: Pola *RF2NVL* i *RF2NVD* są opcjonalne, ale jeśli występują, muszą występować jako para i być sąsiadujące. Para pól może być powtórzona tyle razy, ile jest to wymagane, na przykład:

```
length1 data1 length2 data2 length3 data3
```

Ponieważ te pola są opcjonalne, są pomijane w deklaracjach struktury, które są udostępniane dla różnych obsługiwanych języków programowania.

RF2SID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

RFSIDV,
Identyfikator struktury nagłówka reguł i formatowania.

Wartością początkową tego pola jest RFSIDV.

RF2VER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

RFVER2
Reguły i struktura nagłówka formatowania Version-2 .

Wartością początkową tego pola jest RFVER2.

Wartości początkowe

Tabela 723. Pola w MQRFH2

Nazwa pola	Nazwa stałej	Wartość stałej
RF2SID	RFSIDV,	'RFH¬'
RF2VER	RFVER2	2
RF2LEN	RFLEN2	36
RF2ENC	ENNAT,	Zależy od środowiska
RF2CSI	CINHT	-2
RF2FMT	BRAK FMNONE	Puste
RF2FLG	BRAK RFNONE	0
RF2NVC	Brak	1208

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQRFH2 Structure
D*
D* Structure identifier
D  RF2SID                 1      4    INZ('RFH ')
D* Structure version number
D  RF2VER                 5      8I 0 INZ(2)
D* Total length of MQRFH2 including allNameValueLength and
D* NameValueDatafields
D  RF2LEN                 9     12I 0 INZ(36)
D* Numeric encoding of data that followslast NameValueData field
D  RF2ENC                13     16I 0 INZ(273)
D* Character set identifier of data thatfollows last NameValueData field
D  RF2CSI                17     20I 0 INZ(-2)
D* Format name of data that follows lastNameValueData field
D  RF2FMT                21     28    INZ('        ')
D* Flags
D  RF2FLG                29     32I 0 INZ(0)
D* Character set identifier ofNameValueData
D  RF2NVC                33     36I 0 INZ(1208)
```

IBM i MQRMH (nagłówek komunikatu odniesienia) w systemie IBM i

Struktura MQRMH definiuje format nagłówka komunikatu odniesienia.

Przegląd

Cel: Ten nagłówek jest używany z wyjściami kanału komunikatów zapisanymi przez użytkownika do wysyłania dużych ilości danych (nazywanych "danymi masowymi"). z jednego menedżera kolejek do innego. Różnica w porównaniu do normalnego przesyłania komunikatów polega na tym, że dane masowe nie są przechowywane w kolejce. Zamiast tego w kolejce zapisywane jest tylko *odwołanie* do danych masowych. Zmniejsza to prawdopodobieństwo wyczerpania zasobów IBM MQ przez kilka dużych komunikatów.

Nazwa formatu: FMRMH.

Zestaw znaków i kodowanie: Dane znakowe w MQRMH oraz łańcuchy adresowane przez pola przesunięcia muszą znajdować się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek systemu **CodedCharSetId** . Dane liczbowe w MQRMH muszą być zakodowane na komputerze rodzimym; jest to podawane przez wartość ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQRMH muszą być ustawione w polach *MDCSI* i *MDENC* w następujących polach:

- MQMD (jeśli struktura MQRMH znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQRMH (wszystkie inne przypadki).

Użycie: Aplikacja umieszcza komunikat składający się z MQRMH, ale z pominięciem danych masowych. Gdy komunikat jest odczytywany z kolejki transmisji przez agent kanału komunikatów (MCA), w celu przetworzenia nagłówka komunikatu odniesienia wywoływane jest wyjście komunikatu dostarczone przez użytkownika. Wyjście może dodać do komunikatu odwołania dane masowe identyfikowane przez strukturę MQRMH, zanim agent MCA wyśle komunikat za pośrednictwem kanału do następnego menedżera kolejek.

Na odbierającym końcu powinno istnieć wyjście komunikatu, które oczekuje na komunikaty odniesienia. Po odebraniu komunikatu odniesienia wyjście powinno utworzyć obiekt na podstawie danych masowych, które następują po komunikacie MQRMH, a następnie przekazać komunikat odniesienia bez danych masowych. Komunikat odniesienia może zostać później pobrany przez aplikację, która odczytuje komunikat odniesienia (bez danych masowych) z kolejki.

Zwykle struktura MQRMH jest wszystkim, co znajduje się w komunikacie. Jeśli jednak komunikat znajduje się w kolejce transmisji, jeden lub więcej dodatkowych nagłówków będzie poprzedzających strukturę MQRMH.

Komunikat odniesienia może być również wysłany do listy dystrybucyjnej. W tym przypadku struktura MQDH i powiązane z nią rekordy poprzedzają strukturę MQRMH, gdy komunikat znajduje się w kolejce transmisji.

Uwaga: Komunikat odniesienia nie powinien być wysyłany jako komunikat segmentowany, ponieważ wyjście komunikatu nie może go poprawnie przetworzyć.

- “Konwersja danych” na stronie 1236
- “Pola” na stronie 1237
- “Wartości początkowe” na stronie 1241
- “Deklaracja RPG” na stronie 1242

Konwersja danych

Na potrzeby konwersji danych konwersja struktury MQRMH obejmuje konwersję danych środowiska źródłowego, nazwy obiektu źródłowego, danych środowiska docelowego i nazwy obiektu docelowego. Wszystkie inne bajty w obrębie *RMLEN* bajtów początku struktury są odrzucane lub mają niezdefiniowane wartości po konwersji danych. Dane masowe zostaną przekształcone, pod warunkiem że wszystkie poniższe stwierdzenia są prawdziwe:

- Dane masowe są obecne w komunikacie podczas wykonywania konwersji danych.
- Pole *RMFMT* w MQRMH ma wartość inną niż FMNONE.

- Istnieje zapisane przez użytkownika wyjście konwersji danych o podanej nazwie formatu.

Należy jednak pamiętać, że zazwyczaj dane masowe nie są obecne w komunikacie, gdy komunikat znajduje się w kolejce i że w wyniku tego dane masowe nie będą przekształcane przez opcję GMCONV.

Pola

Struktura MQRMH zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

RMCSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych masowych.

Określa identyfikator zestawu znaków danych masowych; nie ma zastosowania do danych znakowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola *MDPAT* w MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

RMDEL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych środowiska docelowego.

Jeśli to pole ma wartość zero, nie ma danych środowiska docelowego, a parametr *RMDEO* jest ignorowany.

RMDEO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych środowiska docelowego.

To pole określa przesunięcie danych środowiska docelowego od początku struktury MQRMH. Dane środowiska docelowego mogą być określane przez twórcę komunikatu odwołania, jeśli dane te są znane twórcy. Na przykład dane środowiska docelowego mogą być ścieżką do katalogu obiektu, w którym mają być przechowywane dane masowe. Jeśli jednak twórca nie zna danych środowiska docelowego, to za określenie wymaganych informacji o środowisku odpowiada wyjście komunikatu dostarczone przez użytkownika.

Długość danych środowiska docelowego jest określona przez parametr *RMDEL* . Jeśli ta długość wynosi zero, nie ma danych środowiska docelowego, a parametr *RMDEO* jest ignorowany. Jeśli istnieją, dane środowiska docelowego muszą znajdować się całkowicie w obrębie *RMLEN* bajtów od początku struktury.

Aplikacje nie powinny zakładać, że dane środowiska docelowego są ciągłe z danymi adresowanymi przez pola *RMSEO*, *RMSNO*i *RMDNO* .

Wartością początkową tego pola jest 0.

RMDL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych masowych.

Pole *RMDL* określa długość danych masowych, do których odwołuje się struktura MQRMH.

Jeśli w komunikacie znajdują się dane masowe, rozpoczynają się od przesunięcia *RMLEN* bajtów od początku struktury MQRMH. Długość całego komunikatu minus *RMLEN* określa długość danych masowych.

Jeśli w komunikacie znajdują się dane, parametr *RMDL* określa ilość istotnych danych. Normalne jest, że *RMDL* ma taką samą wartość jak długość danych w komunikacie.

Jeśli struktura MQRMH reprezentuje pozostałe dane w obiekcie (począwszy od określonego przesunięcia logicznego), można użyć wartości zero dla parametru *RMDL*, jeśli w komunikacie nie ma danych masowych.

Jeśli nie ma żadnych danych, koniec MQRMH pokrywa się z końcem komunikatu.

Wartością początkową tego pola jest 0.

RMDNL (10-cyfrowa liczba całkowita ze znakiem)

Długość nazwy obiektu docelowego.

Jeśli to pole ma wartość zero, nie ma nazwy obiektu docelowego, a parametr *RMDNO* jest ignorowany.

RMDNO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie nazwy obiektu docelowego.

To pole określa przesunięcie nazwy obiektu docelowego od początku struktury MQRMH. Nazwa obiektu docelowego może zostać określona przez twórcę komunikatu odniesienia, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu docelowego, to do wyjścia komunikatu dostarczonego przez użytkownika należy identyfikowanie obiektu, który ma zostać utworzony lub zmodyfikowany.

Długość nazwy obiektu docelowego jest określona przez parametr *RMDNL* ; Jeśli ta długość wynosi zero, nie ma nazwy obiektu docelowego, a parametr *RMDNO* jest ignorowany. Jeśli istnieje, nazwa obiektu docelowego musi znajdować się w całości w ciągu *RMLEN* bajtów od początku struktury.

Aplikacje nie powinny zakładać, że nazwa obiektu docelowego jest ciągła w połączeniu z danymi adresowanymi przez pola *RMSEO*, *RMSNO*i *RMDEO* .

Wartością początkową tego pola jest 0.

RMDO (10-cyfrowa liczba całkowita ze znakiem)

Małe przesunięcie danych masowych.

To pole określa dolne przesunięcie danych masowych od początku obiektu, którego częścią są dane masowe. Przesunięcie danych masowych od początku obiektu jest nazywane *przesunięciem logicznym*. Nie jest to fizyczne przesunięcie danych masowych od początku struktury MQRMH-to przesunięcie jest nadawane przez *RMLEN*.

Aby umożliwić wysyłanie dużych obiektów przy użyciu komunikatów referencyjnych, przesunięcie logiczne jest podzielone na dwa pola, a rzeczywiste przesunięcie logiczne jest określane przez sumę tych dwóch pól:

- *RMDO* : reszta uzyskana po podzieleniu przesunięcia logicznego przez 1 000 000 000. Jest to zatem wartość z zakresu od 0 do 999 999 999.
- *RMDO2* reprezentuje wynik uzyskany, gdy przesunięcie logiczne zostanie podzielone przez 1 000 000 000. Jest to zatem liczba pełnych wielokrotności 1 000 000 000, które istnieją w przesunięciu logicznym. Liczba wielokrotności mieści się w zakresie od 0 do 999 999 999.

Wartością początkową tego pola jest 0.

RMDO2 (10-cyfrowa liczba całkowita ze znakiem)

Wysokie przesunięcie danych masowych.

To pole określa wysokie przesunięcie danych masowych od początku obiektu, którego częścią są dane masowe. Jest to wartość z zakresu od 0 do 999 999 999. Szczegółowe informacje można znaleźć w sekcji *RMDO*.

Wartością początkową tego pola jest 0.

RMENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe danych masowych.

Określa kodowanie liczbowe danych masowych. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQRMH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest ENNAT.

RMFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi komunikatów odniesienia.

Zdefiniowane są następujące opcje:

RMLAST
Komunikat odniesienia zawiera lub reprezentuje ostatnią część obiektu.

Ta flaga wskazuje, że komunikat odniesienia reprezentuje lub zawiera ostatnią część obiektu odniesienia.

RMNLST,
Komunikat odniesienia nie zawiera ani nie reprezentuje ostatniej części obiektu.

RMNLST jest zdefiniowany w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

Wartością początkową tego pola jest RMNLST.

RMFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych masowych.

Określa nazwę formatu danych masowych.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Wartością początkową tego pola jest FMNONE.

RMLEN (10-cyfrowa liczba całkowita ze znakiem)

Łączna długość MQRMH, w tym łańcuchy na końcu pól stałych, ale bez danych masowych.

Wartością początkową tego pola jest zero.

RMOII (24-bajtowy łańcuch bitowy)

Identyfikator instancji obiektu.

To pole może być używane do identyfikowania konkretnej instancji obiektu. Jeśli nie jest ona potrzebna, należy ją ustawić na następującą wartość:

OIINON
Nie określono identyfikatora instancji obiektu.

Wartością długości pola jest zero binarne.

Długość tego pola jest określona przez LNOIID. Wartością początkową tego pola jest OIINON.

RMOT (8-bajtowy łańcuch znaków)

Typ obiektu.

Jest to nazwa, która może być używana przez wyjście komunikatu do rozpoznawania obsługiwanych przez nie typów komunikatów odniesienia. Należy rozważyć dostosowanie nazwy do tych samych reguł, co pole *RMFMT* .

Wartością początkową tego pola jest 8 odstępów.

RMSEL (10-cyfrowa liczba całkowita ze znakiem)

Długość danych środowiska źródłowego.

Jeśli to pole ma wartość zero, nie ma danych środowiska źródłowego, a parametr *RMSEO* jest ignorowany.

Wartością początkową tego pola jest 0.

RMSEO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie danych środowiska źródłowego.

To pole określa przesunięcie danych środowiska źródłowego od początku struktury MQRMH. Dane środowiska źródłowego mogą być określane przez twórcę komunikatu odniesienia, jeśli są one znane twórcy. Na przykład dane środowiska źródłowego mogą być ścieżką do katalogu obiektu zawierającego dane masowe. Jeśli jednak twórca nie zna danych środowiska źródłowego, to do określenia wymaganych informacji o środowisku należy program zewnętrzny komunikatu dostarczony przez użytkownika.

Długość danych środowiska źródłowego jest określona przez *RMSEL* ; Jeśli ta długość wynosi zero, nie ma danych środowiska źródłowego, a parametr *RMSEO* jest ignorowany. Jeśli są obecne, dane środowiska źródłowego muszą znajdować się całkowicie w ciągu *RMLEN* bajtów od początku struktury.

Aplikacje nie powinny zakładać, że dane środowiska rozpoczynają się bezpośrednio po ostatnim stałym polu w strukturze lub że sąsiadują z danymi adresowanymi przez pola *RMSNO*, *RMDEO*i *RMDNO* .

Wartością początkową tego pola jest 0.

RMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

RMSIDV
Identyfikator struktury nagłówka komunikatu odniesienia.

Wartością początkową tego pola jest RMSIDV.

RMSNL (10-cyfrowa liczba całkowita ze znakiem)

Długość nazwy obiektu źródłowego.

Jeśli to pole ma wartość zero, nie ma nazwy obiektu źródłowego, a parametr *RMSNO* jest ignorowany.

Wartością początkową tego pola jest 0.

RMSNO (10-cyfrowa liczba całkowita ze znakiem)

Przesunięcie nazwy obiektu źródłowego.

To pole określa przesunięcie nazwy obiektu źródłowego od początku struktury MQRMH. Nazwa obiektu źródłowego może być określona przez twórcę komunikatu odniesienia, jeśli dane te są znane twórcy. Jeśli jednak twórca nie zna nazwy obiektu źródłowego, to zadaniem wyjścia komunikatu dostarczonego przez użytkownika jest identyfikowanie obiektu, do którego ma zostać uzyskany dostęp.

Długość nazwy obiektu źródłowego jest określona przez parametr *RMSNL* ; Jeśli ta długość wynosi zero, nie ma nazwy obiektu źródłowego, a parametr *RMSNO* jest ignorowany. Jeśli istnieje, nazwa obiektu źródłowego musi znajdować się w całości w ciągu *RMLEN* bajtów od początku struktury.

Aplikacje nie powinny zakładać, że nazwa obiektu źródłowego jest ciągła w połączeniu z danymi adresowanymi przez pola *RMSEO*, *RMDEO*i *RMDNO* .

Wartością początkową tego pola jest 0.

RMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

RMVER1
Version-1 -struktura nagłówka komunikatu referencyjnego.

Następująca stała określa numer wersji bieżącej:

RRMVERC
Bieżąca wersja struktury nagłówka komunikatu referencyjnego.

Początkową wartością tego pola jest RMVER1.

Wartości początkowe

Tabela 724. Pola w MQRMH

Nazwa pola	Nazwa stałej	Wartość stałej
RMSID	RMSIDV	'RMH¬'
RMVER	RMVER1	1
RMLEN	Brak	0
RMENC	ENNAT,	Zależy od środowiska
RMCSI	CSUNDF	0
RMFMT	BRAK FMNONE	Puste
RMFLG	RMNLST,	0
RMOT	Brak	Puste
RMOII	OIINON	Wartości null
RMSEL	Brak	0
RMSEO	Brak	0
RMSNL	Brak	0
RMSNO	Brak	0
RMDEL	Brak	0
RMDEO	Brak	0
RMDNL	Brak	0
RMDNO	Brak	0
RMDL	Brak	0
RMDO	Brak	0
RMDO2	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
```

```
     D*
     D* MQRMH Structure
     D*
     D* Structure identifier
     D  RMSID                  1      4    INZ('RMH ')
     D* Structure version number
     D  RMVER                  5      8I 0 INZ(1)
     D* Total length of MQRMH, includingstrings at end of fixed fields,but not
     D* the bulk data
     D  RMLEN                  9     12I 0 INZ(0)
     D* Numeric encoding of bulk data
     D  RMENC                 13     16I 0 INZ(273)
     D* Character set identifier of bulkdata
     D  RMCSI                 17     20I 0 INZ(0)
     D* Format name of bulk data
     D  RMFMT                 21     28    INZ('        ')
     D* Reference message flags
     D  RMFLG                 29     32I 0 INZ(0)
     D* Object type
     D  RMOT                  33     40    INZ
     D* Object instance identifier
     D  RMOII                 41     64    INZ(X'00000000000000-
     D                                     00000000000000000000000-
     D                                     000000000000')
     D* Length of source environmentdata
     D  RMSEL                 65     68I 0 INZ(0)
     D* Offset of source environmentdata
     D  RMSEO                 69     72I 0 INZ(0)
     D* Length of source object name
     D  RMSNL                 73     76I 0 INZ(0)
     D* Offset of source object name
     D  RMSNO                 77     80I 0 INZ(0)
     D* Length of destination environmentdata
     D  RMDEL                 81     84I 0 INZ(0)
     D* Offset of destination environmentdata
     D  RMDEO                 85     88I 0 INZ(0)
     D* Length of destination objectname
     D  RMDNL                 89     92I 0 INZ(0)
     D* Offset of destination objectname
     D  RMDNO                 93     96I 0 INZ(0)
     D* Length of bulk data
     D  RMDL                  97    100I 0 INZ(0)
     D* Low offset of bulk data
     D  RMDO                 101    104I 0 INZ(0)
     D* High offset of bulk data
     D  RMDO2                105    108I 0 INZ(0)
```

Deklaracja RPG

IBM i MQRR (rekord odpowiedzi) w systemie IBM i

Struktura MQRR jest używana do odbierania kodu zakończenia i kodu przyczyny będącego wynikiem operacji otwierania lub umieszczania dla pojedynczej kolejki docelowej, gdy miejscem docelowym jest lista dystrybucyjna.

Przegląd

Cel: MQRR jest strukturą wyjściową dla wywołań MQOPEN, MQPUT i MQPUT1 .

Zestaw znaków i kodowanie: Dane w MQRR muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

Użycie: Dzięki udostępnieniu tablicy tych struktur w wywołaniach MQOPEN i MQPUT lub w wywołaniu MQPUT1 można określić kody zakończenia i kody przyczyny dla wszystkich kolejek na liście dystrybucyjnej, gdy wynik wywołania jest mieszany, czyli gdy wywołanie powiedzie się dla niektórych kolejek na liście, ale zakończy się niepowodzeniem dla innych kolejek. Kod przyczyny RC2136 z wywołania wskazuje, że rekordy odpowiedzi (jeśli zostały udostępnione przez aplikację) zostały ustawione przez menedżer kolejek.

- “Pola” na stronie 1243

- “Wartości początkowe” na stronie 1243
- “Deklaracja RPG” na stronie 1243

Pola

Struktura MQRR zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

RRCC (10-cyfrowa liczba całkowita ze znakiem)

Kod zakończenia dla kolejki.

Jest to kod zakończenia wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR udostępnionej w wywołaniu MQOPEN lub MQPUT1 .

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest CCOK.

RRREA (10-cyfrowa liczba całkowita ze znakiem)

Kod przyczyny dla kolejki.

Jest to kod przyczyny wynikający z operacji otwierania lub umieszczania dla kolejki o nazwie określonej przez odpowiedni element w tablicy struktur MQOR podanej w wywołaniu MQOPEN lub MQPUT1 .

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest RCNONE.

Wartości początkowe

Tabela 725. Pola w MQRR

Nazwa pola	Nazwa stałej	Wartość stałej
RRCC	CKOK	0
RRREA	BRAK RCNONE	0

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQRR Structure
D*
D* Completion code for queue
D  RRCC                  1      4I 0 INZ(0)
D* Reason code for queue
D  RRREA                 5      8I 0 INZ(0)
```

IBM i MQSCO (opcje konfiguracyjne TLS) w systemie IBM i

Struktura MQSCO (z polami TLS w strukturze MQCD) umożliwia aplikacji działającej jako IBM MQ MQI client określenie opcji konfiguracyjnych, które sterują użyciem protokołu TLS dla połączenia klienckiego, gdy protokołem kanału jest TCP/IP.

Przegląd

Cel: Struktura jest parametrem wejściowym wywołania MQCONNX.

Jeśli protokołem kanału klienta nie jest TCP/IP, struktura MQSCO jest ignorowana.

Zestaw znaków i kodowanie: dane w MQSCO muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT.

Pola

Struktura MQSCO zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

SCAIC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba rekordów informacji uwierzytelniających (MQAIR) zaadresowanych w polach *SCAIP* lub *SCAIO* . Więcej informacji na ten temat zawiera sekcja“MQAIR (rekord informacji uwierzytelniającej) w systemie IBM i” na stronie 1042. Wartość musi być równa zero lub większa. Jeśli wartość nie jest poprawna, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2383.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

SCAIO (10-cyfrowa liczba całkowita ze znakiem)

Jest to przesunięcie w bajtach pierwszego rekordu informacji uwierzytelniającej od początku struktury MQSCO. Przesunięcie może być dodatnie lub ujemne. Pole jest ignorowane, jeśli parametr *SCAIC* ma wartość zero.

Do określenia rekordów MQAIR można użyć wartości *SCAIO* lub *SCAIP* , ale nie obu tych wartości. Szczegółowe informacje można znaleźć w opisie pola *SCAIP* .

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

SCAIP (10-cyfrowa liczba całkowita ze znakiem)

Jest to adres pierwszego rekordu informacji uwierzytelniającej. Pole jest ignorowane, jeśli parametr *SCAIC* ma wartość zero.

Tablicę rekordów MQAIR można udostępnić na jeden z dwóch sposobów:

- Za pomocą pola wskaźnika *SCAIP*

  W takim przypadku aplikacja może zadeklarować tablicę rekordów MQAIR, która jest oddzielona od struktury MQSCO, i ustawić parametr *SCAIP* na adres tablicy.

  Należy rozważyć użycie języka *SCAIP* dla języków programowania obsługujących typ danych wskaźnika w sposób, który jest przenośny w różnych środowiskach (na przykład w języku programowania C).

- Używając pola przesunięcia *SCAIO*

  W takim przypadku aplikacja musi zadeklarować strukturę złożoną zawierającą obiekt MQSCO, po której następuje tablica rekordów MQAIR, i ustawić parametr *SCAIO* na przesunięcie pierwszego rekordu w tablicy od początku struktury MQSCO. Upewnij się, że ta wartość jest poprawna i ma wartość, która może być ustawiona w MQLONG (najbardziej restrykcyjnym językiem programowania jest COBOL, dla którego poprawny zakres to od -999 999 999 do +999 999 999).

  Należy rozważyć użycie języka *SCAIO* dla języków programowania, które nie obsługują typu danych wskaźnika lub implementują typ danych wskaźnika w sposób, który nie jest przenośny w różnych środowiskach (na przykład w języku programowania COBOL).

Niezależnie od wybranej techniki można użyć tylko jednej z metod *SCAIP* i *SCAIO* ; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2384 , jeśli obie są niezerowe.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty w tych językach programowania, które obsługują wskaźniki, a w przeciwnym razie jest to łańcuch bajtowy o wartości all-null.

Uwaga: Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

SCCERLBL (10-cyfrowa liczba całkowita ze znakiem)

W tym polu znajdują się szczegółowe informacje na temat używanej etykiety certyfikatu.

IBM MQ inicjuje pole SCCERLBL jako puste. Wprowadź wymaganą wartość lub zaakceptuj wartość domyślną.

`ibmwebspheremq`*`user_id`* jest poprawną wartością dla tego pola dla wszystkich wersji produktu, a dla MQSCO w wersji wcześniejszej niż 5.0 jest jedyną poprawną wartością. Dlatego wartość tego pola jest interpretowana w czasie wykonywania i w razie potrzeby zmieniana. Jeśli zostanie podana opcja MQSCO w wersji wcześniejszej niż 5.0lub zostanie zaakceptowana domyślna wartość odstępu w polu SCCERLBL, system użyje wartości `ibmwebspheremq`*`user_id`*.

Jest to pole wejściowe.

SCCERTVPOL (10-cyfrowa liczba całkowita ze znakiem)

To pole określa, jaki typ strategii sprawdzania poprawności certyfikatu jest używany. Pole można ustawić na jedną z następujących wartości:

MQ_CERT_VAL_POLICY_ANY
Zastosuj wszystkie strategie sprawdzania poprawności certyfikatów obsługiwane przez bibliotekę bezpiecznych gniazd. Zaakceptuj łańcuch certyfikatów, jeśli dowolna strategia uzna, że łańcuch certyfikatów jest poprawny.

MQ_CERT_VAL_POLICY_RFC5280
Zastosuj tylko strategię sprawdzania poprawności certyfikatu zgodną ze standardem RFC5280 . To ustawienie zapewnia bardziej rygorystyczne sprawdzanie poprawności niż ustawienie ANY, ale odrzuca niektóre starsze certyfikaty cyfrowe.

Wartością początkową tego pola jest MQ_CERT_VAL_POLICY_ANY

SCCH (10-cyfrowa liczba całkowita ze znakiem)

To pole zawiera szczegóły konfiguracji sprzętu szyfrującego połączonego z systemem klienckim.

Ustaw pole na łańcuch w następującym formacie lub pozostaw je puste lub puste:

```
GSK_PKCS11=the PKCS #11 driver path and file name;the PKCS #11
token label;the PKCS #11 token password;symmetric cipher setting>;
```

Aby użyć sprzętu szyfrującego, który jest zgodny z interfejsem PKCS11 , na przykład IBM 4960 lub IBM 4963, podaj ścieżkę sterownika PKCS11 , etykietę tokenu PKCS11 i łańcuch hasła tokenu PKCS11 , każdy zakończony średnikiem.

Ścieżka sterownika PKCS #11 jest pełną ścieżką do biblioteki współużytkowanej zapewniającej obsługę karty PKCS #11 . Nazwa pliku sterownika PKCS #11 jest nazwą biblioteki współużytkowanej. Przykład wartości wymaganej dla ścieżki PKCS #11 i nazwy pliku:

```
/usr/lib/pkcs11/PKCS11_API.so
```

Etykieta tokenu PKCS #11 musi być zapisana małymi literami. Jeśli sprzęt został skonfigurowany z użyciem etykiety z literami o różnej wielkości lub wielkimi literami, należy zmienić jego konfigurację przy użyciu tej etykiety z małymi literami.

Jeśli nie jest wymagana żadna konfiguracja sprzętu szyfrującego, należy ustawić to pole na wartość pustą lub pustą.

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Jeśli wartość nie jest poprawna lub prowadzi do awarii podczas konfigurowania sprzętu szyfrującego, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2382.

Jest to pole wejściowe. Długość tego pola jest określona przez LNSSCH. Wartość początkowa tego pola jest pusta.

SCEPSUITEB (10-cyfrowa liczba całkowita ze znakiem)
To pole określa, czy używane jest szyfrowanie zgodne ze standardem Suite B oraz jaki poziom mocy jest stosowany. Wartość może być jedną lub kilkoma z następujących wartości:

- SCEPSUITEB0

  Szyfrowanie zgodne ze standardem Suite B nie jest używane.

- SCEPSUITEB1

  Używany jest 128-bitowy poziom bezpieczeństwa standardu Suite B.

- SCEPSUITEB2

  Używany jest 192-bitowy poziom bezpieczeństwa standardu Suite B.

Uwaga: Użycie opcji SCEPSUITEB0 z dowolną inną wartością w tym polu jest niepoprawne.

SCFR (10-cyfrowa liczba całkowita ze znakiem)

Produkt IBM MQ można skonfigurować z użyciem sprzętu szyfrującego w taki sposób, aby używane moduły kryptograficzne były modułami dostarczanymi przez produkt sprzętowy. Moduły te mogą mieć certyfikat FIPS na określonym poziomie w zależności od używanego produktu.

To pole służy do określenia, że używane są tylko algorytmy z certyfikatem FIPS, jeśli szyfrowanie jest udostępniane w oprogramowaniu dostarczonym przez IBM MQ.

Po zainstalowaniu produktu IBM MQ instalowana jest również implementacja szyfrowania TLS, która udostępnia niektóre moduły z certyfikatem FIPS.

Wartości mogą być następujące:

MQSSL_FIPS_NO
Jest to wartość domyślna. W przypadku ustawienia tej wartości:

- Można użyć dowolnej CipherSpec obsługiwanej na konkretnej platformie.
- Jeśli produkt jest uruchamiany bez użycia sprzętu szyfrującego, następujące CipherSpecs są uruchamiane z użyciem szyfrowania z certyfikatem FIPS 140-2 na platformach IBM 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
W przypadku ustawienia tej wartości, o ile do szyfrowania nie jest używany sprzęt szyfrujący, można mieć pewność, że

- W specyfikacji szyfrowania CipherSpec mającej zastosowanie do tego połączenia klienta mogą być używane tylko algorytmy szyfrowania z certyfikatem FIPS.
- Przychodzące i wychodzące połączenia kanału TLS powiodą się tylko wtedy, gdy używana jest jedna z następujących specyfikacji szyfru:
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
  - TLS_RSA_WITH_AES_128_CBC_SHA
  - TLS_RSA_WITH_AES_256_CBC_SHA

Uwagi:

1. Deprecated CipherSpec TLS_RSA_WITH_3DES_EDE_CBC_SHA jest nieaktualna.
2. Tam, gdzie to możliwe, jeśli skonfigurowano CipherSpecs tylko dla FIPS, klient MQI odrzuca połączenia, które określają specyfikację szyfrowania CipherSpec withRC2393. IBM MQ nie gwarantuje odrzucenia wszystkich takich połączeń i jest odpowiedzialny za określenie, czy konfiguracja IBM MQ jest zgodna ze standardami FIPS.

V 9.3.0 V 9.3.0 **SCKEYPWL (10-cyfrowa liczba całkowita ze znakiem)**

Jest to długość frazy hasła repozytorium kluczy TLS.

Maksymalna długość hasła repozytorium kluczy wynosi 128 znaków. Jeśli fraza hasła repozytorium kluczy jest większa niż maksymalna dozwolona długość, połączenie kończy się niepowodzeniem z kodem powrotu RC2381.

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

V 9.3.0 V 9.3.0 **SCKEYPWO (10-cyfrowa liczba całkowita ze znakiem)**

Jest to przesunięcie w bajtach hasła repozytorium kluczy TLS. Przesunięcie może być dodatnie lub ujemne.

Do określenia frazy hasła repozytorium kluczy można użyć wartości SCKEYPWO lub SCKEYPWP , ale nie obu tych wartości. Więcej informacji na ten temat zawiera opis pola SCKEYPWP .

Jest to pole wejściowe. Wartością początkową tego pola jest 0.

V 9.3.0 V 9.3.0 **SCKEYPWP (wskaźnik)**

Jest to adres frazy hasła repozytorium kluczy TLS.

Jest to pole wejściowe. Wartością początkową tego pola jest wskaźnik pusty.

Frazę hasła repozytorium kluczy można podać w postaci jawnego łańcucha tekstowego lub frazę hasła, która została zaszyfrowana za pomocą programu narzędziowego **runmqicred** .

Fraza hasła repozytorium kluczy podana w tym polu nadpisuje wszystkie frazy hasła repozytorium kluczy podane w zmiennej środowiskowej *MQKEYRPWD* lub we właściwości *SSLKeyRepositoryPassword* w sekcji SSL pliku konfiguracyjnego klienta.

Do określenia frazy hasła repozytorium kluczy można użyć wartości SCKEYPWO lub SCKEYPWP , ale nie obu tych wartości.

SCKR (10-cyfrowa liczba całkowita ze znakiem)

To pole określa położenie pliku bazy danych kluczy, w którym przechowywane są klucze i certyfikaty.

V 9.3.0 V 9.3.0 Jeśli przyrostek pliku nie zostanie określony, automatycznie dodawany jest przyrostek `.kdb` .

Z każdym plikiem bazy danych kluczy może być skojarzony *plik ukrytych haseł*. Zawiera zaszyfrowane hasła, które są używane do uzyskania programowego dostępu do bazy danych kluczy. Plik ukrytych haseł musi znajdować się w tym samym katalogu i mieć ten sam rdzeń, co baza danych kluczy, i musi kończyć się przyrostkiem `.sth`.

Jeśli na przykład plik bazy danych kluczy to `/xxx/yyy/key.kdb`, plik ukrytych haseł musi mieć nazwę `/xxx/yyy/key.sth`, gdzie `xxx` i `yyy` reprezentują nazwy katalogów.

V 9.3.0 V 9.3.0 Hasło bazy danych kluczy można również określić w polach *SCKEYPWP* lub *SCKEYPWO* .

Jeśli wartość jest krótsza niż długość pola, należy zakończyć wartość znakiem o kodzie zero lub dopełnić ją spacjami do długości pola. Wartość nie jest sprawdzana. Jeśli wystąpi błąd podczas uzyskiwania dostępu do repozytorium kluczy, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2381.

Aby uruchomić połączenie TLS z serwera IBM MQ MQI client, należy ustawić parametr *SCKR* na poprawną nazwę pliku bazy danych kluczy.

Jest to pole wejściowe. Długość tego pola jest określona przez LNSSKR. Wartość początkowa tego pola jest znakiem odstępu.

SCSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury; wartość musi być następująca:

SCSIDV
Identyfikator struktury opcji konfiguracyjnych TLS.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest SCSIDV.

SCVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

SCVER1
Version-1 Struktura opcji konfiguracyjnych TLS.

SCVER2
Version-2 Struktura opcji konfiguracyjnych TLS.

SCVER3
Version-3 Struktura opcji konfiguracyjnych TLS.

SCVER4
Version-4 Struktura opcji konfiguracyjnych TLS.

SCVER5
Version-5 Struktura opcji konfiguracyjnych TLS.

SCVER6
Version-6 Struktura opcji konfiguracyjnych TLS.

Następująca stała określa numer wersji bieżącej:

SCVERC
Bieżąca wersja struktury opcji konfiguracyjnych TLS.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest SCVER1.

Wartości początkowe

Tabela 726. Pola w MQSCO

Nazwa pola	Nazwa stałej	Wartość stałej
SCSID	SCSIDV	'SCO¬'
SCVER	SCVER1	1
SCKR	Brak	Pusty łańcuch lub odstępy
SCCH	Brak	Pusty łańcuch lub odstępy
SCAIC	Brak	0
SCAIO	Brak	0
SCAIP	Brak	Pusty wskaźnik lub puste bajty
SCKRC	Brak	Pusty wskaźnik lub puste bajty
SCFR	Brak	Pusty wskaźnik lub puste bajty
SCEPSUITEB	Brak	Pusty wskaźnik lub puste bajty
SCCERTVPOL	Brak	Pusty wskaźnik lub puste bajty

Tabela 726. Pola w MQSCO (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
SCCERLBL	Brak	Pusty wskaźnik lub puste bajty
V 9.3.0 V 9.3.0 *SCKEYPWP*	Brak	Pusty wskaźnik lub puste bajty
V 9.3.0 V 9.3.0 *SCKEYPWO*	Brak	0
V 9.3.0 V 9.3.0 *SCKEYPWL*	Brak	0
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu. 2. Informacje na temat opcji SCEPSUITEB zawiera sekcja “Deklaracja RPG” na stronie 1249 .		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSCO Structure
D*
D* Structure identifier
D  SCSID                  1      4    INZ('SCO ')
D* Structure version number
D  SCVER                  5      8I 0 INZ(1)
D* Location of TLS key repository
D  SCKR                   9    264    INZ
D* Cryptographic hardware configuration string
D  SCCH                 265    520    INZ
D* Number of MQAIR records present
D  SCAIC                521    524I 0 INZ(0)
D* Offset of first MQAIR record from start of MQSCO structure
D  SCAIO                525    528I 0 INZ(0)
D* Address of first MQAIR record
D SCAIP                 529    544*   INZ(*NULL)
D* Ver:1 **
D* Number of unencrypted bytes sent/received before secret key is
D* reset
D SCKRC                 545    548I 0 INZ(0)
D* Using FIPS-certified algorithms
D SCFR                  549    552I 0 INZ(0)
D* Ver:2 **
* Use only Suite B cryptographic algorithms
D SCEPSUITEB0
D SCEPSUITEB1           553    556I 0 INZ(1)
D SCEPSUITEB2           557    560I 0 INZ(0)
D SCEPSUITEB3           561    564I 0 INZ(0)
D SCEPSUITEB4           565    568I 0 INZ(0)
D SCEPSUITEB                    10I 0 DIM(4) OVERLAY(SCEPSUITEB0)
D* Ver:3 **
D* Certificate validation policy
D SCCERTVPOL            569    572I 0 INZ(0)
D* Ver:4 **
```

IBM i MQSD (deskryptor subskrypcji) w systemie IBM i

Struktura MQSD służy do określania szczegółów dotyczących budowanej subskrypcji.

Przegląd

Przeznaczenie
Struktura jest parametrem wejścia/wyjścia w wywołaniu MQSUB.

Subskrypcje zarządzane
Jeśli aplikacja nie musi używać określonej kolejki jako miejsca docelowego dla tych publikacji, które są zgodne z jej subskrypcją, może użyć funkcji subskrypcji zarządzanej. Jeśli aplikacja zdecyduje się na użycie subskrypcji zarządzanej, menedżer kolejek poinformuje subskrybenta o miejscu docelowym, do którego są wysyłane opublikowane komunikaty, udostępniając uchwyt obiektu jako dane wyjściowe wywołania MQSUB. Więcej informacji na ten temat zawiera sekcja HOBJ (10-cyfrowa liczba całkowita ze znakiem)-input/output.

Po usunięciu subskrypcji menedżer kolejek zobowiązuje się również do czyszczenia komunikatów, które nie zostały pobrane z zarządzanego miejsca docelowego, w następujących sytuacjach:

- Po usunięciu subskrypcji-za pomocą komendy MQCLOSE z mechanizmem CORMSB-i zamknięciu zarządzanego urządzenia Hobj.
- W sposób niejawny oznacza utratę połączenia z aplikacją korzystającą z subskrypcji nietrwałej (SONDUR).
- Po utracie ważności, gdy subskrypcja jest usuwana, ponieważ utraciła ważność, a zarządzany obiekt Hobj jest zamknięty.

Należy użyć subskrypcji zarządzanych z subskrypcjami nietrwałymi, aby procedura czyszcząca mogła zostać wykonana, a komunikaty dla zamkniętych subskrypcji nietrwałych nie zajmą miejsca w menedżerze kolejek. Trwałe subskrypcje mogą również używać zarządzanych miejsc docelowych.

Zestaw znaków i kodowanie
Dane w usłudze MQSD muszą być w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez translację ENNAT. Jeśli jednak aplikacja działa jako klient IBM MQ , struktura musi być w zestawie znaków i kodowaniu klienta.

- "Pola" na stronie 1250
- "Wartości początkowe" na stronie 1263
- "Deklaracja RPG" na stronie 1264

Pola

Struktura MQSD zawiera następujące pola; pola są opisane w porządku alfabetycznym:

SDAID (32-bajtowy łańcuch znaków)

Ta wartość znajduje się w polu *MDAID* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *SDAID* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

Więcej informacji na temat *MDAID* zawiera sekcja MDAID.

Jeśli opcja S0SETI nie jest określona, parametr *MDAID* , który jest ustawiany w każdym komunikacie publikowanym dla tej subskrypcji, jest pusty jako domyślna informacja o kontekście.

Jeśli podano opcję S0SETI , plik *SDAID* jest generowany przez użytkownika, a to pole jest polem wejściowym zawierającym parametr *MDAID* , który ma zostać ustawiony w każdej publikacji dla tej subskrypcji.

Długość tego pola jest określona przez LNAIDD. Wartość początkowa tego pola to 32 znaki odstępu.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji SOALT można zmienić wartość parametru *SDAID* dla wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB korzystającego z usługi SORES w tym polu jest ustawiana bieżąca wartość parametru *MDAID* używanego dla subskrypcji.

SDACC (32 bajtowy łańcuch znaków)

Ta wartość znajduje się w polu *MDACC* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. *MDACC* jest częścią kontekstu tożsamości komunikatu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu.

Więcej informacji na temat *MDACC* zawiera sekcja MDACC.

W polu *SDACC* można użyć następującej wartości specjalnej:

ACNONE (brak)
Nie określono tokenu rozliczania.

Wartością długości pola jest zero binarne.

Jeśli opcja S0SETI nie jest określona, znacznik rozliczania jest generowany przez menedżer kolejek jako domyślne informacje o kontekście, a to pole jest polem wyjściowym zawierającym element *MDACC* , który jest ustawiany w każdym komunikacie publikowanym dla tej subskrypcji.

Jeśli podano opcję S0SETI , token rozliczania jest generowany przez użytkownika, a to pole jest polem wejściowym zawierającym parametr *MDACC* , który ma zostać ustawiony w każdej publikacji dla tej subskrypcji.

Długość tego pola jest określona przez LNACCT. Wartością początkową tego pola jest ACNONE.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji S0ALT wartość parametru *MDACC* w przyszłych komunikatach publikacji może zostać zmieniona.

Po powrocie z wywołania MQSUB używającego S0RESw tym polu jest ustawiana bieżąca wartość *MDACC* używana dla subskrypcji.

SDASI (40-bajtowy łańcuch bitowy)

Jest to identyfikator bezpieczeństwa, który jest przekazywany z *SDAU* do usługi autoryzacji w celu umożliwienia przeprowadzenia odpowiednich sprawdzeń autoryzacji.

Parametr *SDASI* jest używany tylko wtedy, gdy podano wartość S0ALTU , a pole *SDAU* nie jest całkowicie puste aż do pierwszego znaku o kodzie zero lub końca pola.

W przypadku powrotu z wywołania MQSUB używającego S0RESto pole pozostaje niezmienione.

Więcej informacji na ten temat zawiera opis funkcji ODASI w typie danych MQOD.

SDAU (12-bajtowy łańcuch znaków)

Jeśli zostanie podana wartość S0ALTU, to pole zawiera alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji subskrypcji i danych wyjściowych do kolejki docelowej (określonej w parametrze **Hobj** wywołania MQSUB) zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona.

Jeśli operacja powiedzie się, identyfikator użytkownika określony w tym polu jest rejestrowany jako identyfikator użytkownika będącego właścicielem subskrypcji zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona.

Jeśli określono wartość S0ALTU i pole to jest całkowicie puste, aż do pierwszego znaku o kodzie zero lub końca pola, subskrypcja może zakończyć się powodzeniem tylko wtedy, gdy do subskrybowania tego tematu z podanymi opcjami lub z kolejką docelową dla danych wyjściowych nie jest wymagana autoryzacja użytkownika.

Jeśli wartość S0ALTU nie jest określona, to pole jest ignorowane.

W przypadku powrotu z wywołania MQSUB używającego S0RESto pole pozostaje niezmienione.

Jest to pole wejściowe. Długość tego pola jest określona przez LNUID. Wartością początkową tego pola jest 12 pustych znaków.

SDCID (24-bajtowy łańcuch)

Wszystkie publikacje wysłane w celu dopasowania tej subskrypcji zawierają ten identyfikator korelacji w deskryptorze komunikatu. Jeśli wiele subskrypcji używa tej samej kolejki do pobierania swoich publikacji, użycie komendy MQGET według identyfikatora korelacji umożliwia uzyskanie tylko publikacji dla konkretnej subskrypcji. Ten identyfikator korelacji może zostać wygenerowany przez menedżera kolejek lub przez użytkownika.

Jeśli opcja SOSCID nie jest określona, identyfikator korelacji jest generowany przez menedżer kolejek, a to pole jest polem wyjściowym zawierającym identyfikator korelacji ustawiony w każdym komunikacie publikowanym dla tej subskrypcji.

Jeśli podano opcję SOSCID , identyfikator korelacji jest generowany przez użytkownika, a to pole jest polem wejściowym zawierającym identyfikator korelacji, który ma zostać ustawiony w każdej publikacji dla tej subskrypcji. W tym przypadku, jeśli pole zawiera CINONE, identyfikator korelacji, który jest ustawiany w każdym komunikacie publikowanym dla tej subskrypcji, jest identyfikatorem korelacji utworzonym przez oryginalne umieszczenie komunikatu.

Jeśli określono opcję SOGRP , a określony identyfikator korelacji jest taki sam jak istniejąca zgrupowana subskrypcja używająca tej samej kolejki i nakładającego się łańcucha tematu, z kopią publikacji jest udostępniana tylko najbardziej znacząca subskrypcja w grupie.

Długość tego pola jest określona przez LNCID. Wartością początkową tego pola jest CINONE.

Jeśli istniejąca subskrypcja jest zmieniana przy użyciu opcji SOALT , a to pole jest polem wejściowym, można zmienić identyfikator korelacji subskrypcji, chyba że subskrypcja została utworzona przy użyciu opcji SOGRP .

W przypadku powrotu z wywołania MQSUB z użyciem opcji SORESw tym polu jest ustawiany bieżący identyfikator korelacji dla subskrypcji.

SDEXP (10-cyfrowa liczba całkowita ze znakiem)

Jest to czas wyrażony w dziesiątych częściach sekundy, po upływie którego subskrypcja traci ważność. Po upływie tego okresu żadne publikacje nie będą zgodne z tą subskrypcją. Jest ona również używana jako wartość w polu *MDEXP* deskryptora MQMD publikacji wysłanych do tego subskrybenta.

Rozpoznawana jest następująca wartość specjalna:

EIULIM
Subskrypcja ma nieograniczony czas ważności.

W przypadku zmiany istniejącej subskrypcji przy użyciu opcji SOALT można zmienić termin ważności subskrypcji.

Po powrocie z wywołania MQSUB z użyciem opcji SORES w tym polu jest ustawiana pierwotna utrata ważności subskrypcji, a nie pozostały czas utraty ważności.

SDON (48-bajtowy łańcuch znaków)

Jest to nazwa obiektu tematu zdefiniowana w lokalnym menedżerze kolejek.

Nazwa może zawierać następujące znaki:

- Wielkie litery (od A do Z)
- Małe litery (od a do z)
- Cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani wewnętrznych odstępów, ale może zawierać końcowe odstępy. Znak o kodzie zero oznacza koniec istotnych danych w nazwie; znak o kodzie zero i wszystkie następujące po nim znaki są traktowane jako odstępy. Zastosowanie mają następujące ograniczenia:

- W systemach używających kodu EBCDIC Katakana nie można używać małych liter.

- Nazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w cudzysłów, jeśli zostały podane w komendach. Te cudzysłowy nie mogą być podawane dla nazw, które występują jako pola w strukturach lub jako parametry w wywołaniach.

Nazwa *SDON* jest używana do tworzenia pełnej nazwy tematu.

Pełną nazwę tematu można utworzyć na podstawie dwóch różnych pól: *SDON* i *SDOS*. Szczegółowe informacje na temat używania tych dwóch pól zawiera sekcja Łączenie łańcuchów tematów.

Po powrocie z wywołania MQSUB z opcją S0RES to pole pozostaje niezmienione.

Długość tego pola jest określona przez LNTOPN. Wartością początkową tego pola jest 48 znaków odstępu.

W przypadku zmiany istniejącej subskrypcji przy użyciu opcji SDALT nie można zmienić nazwy obiektu tematu, który został zasubskrybowany. To pole i pole *SDOS* można pominąć. Jeśli zostaną podane, muszą zostać przetłumaczone na tę samą pełną nazwę tematu, w przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2510 .

SDOPT (10-cyfrowa liczba całkowita ze znakiem)

Należy podać co najmniej jedną z następujących opcji:

- SOALT (SOALT)
- SYSTEMY SORES
- SOCRT

Wartości można dodawać. Nie należy dodawać tej samej stałej więcej niż raz. W tabeli przedstawiono, w jaki sposób można połączyć te opcje: połączenia, które nie są poprawne, są odnotowywane; wszystkie inne kombinacje są poprawne.

Opcje dostępu lub tworzenia

Opcje dostępu i tworzenia określają, czy subskrypcja jest tworzona, czy też istniejąca subskrypcja jest zwracana lub zmieniana. Należy określić co najmniej jedną z tych opcji. Tabela zawiera poprawne kombinacje opcji dostępu lub tworzenia.

Tabela 727. Poprawne kombinacje opcji dostępu i tworzenia

Kombinacja opcji	**Uwagi**
SOCRT	Tworzy subskrypcję, jeśli nie istnieje; kończy się niepowodzeniem, jeśli subskrypcja istnieje.
SRES	Wznawia istniejącą subskrypcję. Niepowodzenie, jeśli nie istnieje żadna subskrypcja.
SOCRT + SORES	Tworzy subskrypcję, jeśli taka subskrypcja nie istnieje, i wznawia ją, jeśli taka subskrypcja istnieje. Przydatna kombinacja, jeśli jest używana w aplikacji, która może być uruchamiana wiele razy.
SORES + SOALT (patrz uwaga)	Wznawia istniejącą subskrypcję, zmieniając pola w taki sposób, aby były zgodne z polami określonymi w pliku MQSD, kończy się niepowodzeniem, jeśli nie istnieje żadna subskrypcja.

Tabela 727. Poprawne kombinacje opcji dostępu i tworzenia (kontynuacja)	
Kombinacja opcji	**Uwagi**
SOCRT + SOALT (patrz uwaga)	Tworzy subskrypcję, jeśli taka subskrypcja nie istnieje, i wznawia zgodną subskrypcję, jeśli taka subskrypcja istnieje, zmieniając dowolne pola w taki sposób, aby były zgodne z polami określonymi w pliku MQSD. Przydatna kombinacja, jeśli jest używana w aplikacji, która przed kontynuowaniem chce upewnić się, że jej subskrypcja jest w określonym stanie.

Uwaga:

Opcje określające S0ALT mogą również określać S0RES, ale ta kombinacja nie ma dodatkowego wpływu na określanie samej wartości S0ALT . S0ALT oznacza S0RES, ponieważ wywołanie MQSUB w celu zmiany subskrypcji oznacza, że subskrypcje również zostały wznowione. Sytuacja odwrotna nie jest prawdziwa, jednak wznowienie subskrypcji nie oznacza, że należy ją zmienić.

SOCRT

Utwórz subskrypcję dla określonego tematu. Jeśli istnieje subskrypcja używająca tego samego źródła danych *SDSN* , wywołanie kończy się niepowodzeniem z błędem RC2432 . Tego niepowodzenia można uniknąć, łącząc opcję S0CRT z opcją S0RES. *SDSN* nie zawsze jest konieczne. Więcej informacji na ten temat zawiera opis tego pola.

Połączenie komendy S0CRT z opcją S0RES najpierw sprawdza, czy istnieje subskrypcja dla określonego elementu *SDSN*i czy do tej istniejącej subskrypcji jest zwracany uchwyt. Jeśli jednak nie ma istniejącej subskrypcji, tworzona jest nowa subskrypcja przy użyciu wszystkich pól podanych w pliku MQSD.

W celu uzyskania podobnego efektu można również połączyć S0CRT z S0ALT (szczegółowe informacje na temat komendy S0ALT znajdują się w dalszej części tego tematu).

SRES

Zwraca uchwyt do istniejącej subskrypcji, która jest zgodna z tymi określonymi w parametrze *SDSN*. W zgodnych atrybutach subskrypcji nie są wprowadzane żadne zmiany i są one zwracane w danych wyjściowych w strukturze MQSD. Większość treści usługi MQSD nie jest używana: używane są pola *SDSID*, *SDVER*, *SDOPT*, *SDAID* i *SDASI*oraz *SDSN*.

Wywołanie kończy się niepowodzeniem z kodem przyczyny RC2428 , jeśli subskrypcja nie istnieje i nie jest zgodna z pełną nazwą subskrypcji. Tego niepowodzenia można uniknąć, łącząc opcję S0CRT z opcją S0RES. Szczegółowe informacje na temat komendy S0CRTzawiera sekcja SOCRT.

Identyfikator użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub, jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika ostatniej pomyślnej zmiany. Jeśli używana jest wartość *SDAID* i dla tego użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkowników, wartość *SDAID* jest rejestrowana jako identyfikator użytkownika, który utworzył subskrypcję, a nie jako identyfikator użytkownika, w ramach którego została utworzona subskrypcja.

Identyfikator użytkownika, który utworzył subskrypcję, jest rejestrowany jako *SDAU* , jeśli to pole jest używane, a dla tego użytkownika dozwolone jest użycie alternatywnych identyfikatorów użytkownika.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji S0AUID , a identyfikator użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu subskrypcji, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2434 .

Jeśli zgodna subskrypcja istnieje i jest obecnie używana przez inną aplikację, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2429 . Jeśli jest ona obecnie używana przez to samo połączenie, wywołanie nie kończy się niepowodzeniem i zwracany jest uchwyt do subskrypcji.

Jeśli subskrypcja o nazwie określonej w polu SubName nie jest poprawną subskrypcją, która może zostać wznowiona lub zmieniona w aplikacji, wywołanie kończy się niepowodzeniem z błędem RC2523 .

S0RES jest implikowany przez S0ALT i dlatego nie jest wymagane łączenie z tą opcją, jednak nie jest to błąd, jeśli te dwie opcje są połączone.

SOALT

Zwraca uchwyt do istniejącej subskrypcji, której pełna nazwa jest zgodna z nazwą określoną w parametrze *SDSN*. Wszystkie atrybuty subskrypcji, które różnią się od atrybutów określonych w pliku MQSD, są zmieniane w subskrypcji, chyba że zmiana jest niedozwolona dla tego atrybutu. Szczegóły znajdują się w opisie każdego atrybutu i są podsumowane w poniższej tabeli. Próba zmiany atrybutu, którego nie można zmienić, zakończy się niepowodzeniem z kodem przyczyny przedstawionym w poniższej tabeli.

Wywołanie kończy się niepowodzeniem z kodem przyczyny RC2428 , jeśli subskrypcja nie istnieje i nie jest zgodna z pełną nazwą subskrypcji. Tego niepowodzenia można uniknąć, łącząc opcję S0CRT z opcją S0ALT.

Połączenie komendy S0CRT z subskrypcją S0ALT najpierw sprawdza, czy istnieje subskrypcja dla określonej pełnej nazwy subskrypcji i czy zwraca uchwyt do tej wcześniej istniejącej subskrypcji ze zmianami wprowadzonymi zgodnie z wcześniejszymi instrukcjami, ale jeśli nie ma istniejącej subskrypcji, tworzona jest nowa subskrypcja przy użyciu wszystkich pól podanych w tabeli MQSD.

Identyfikator użytkownika subskrypcji jest identyfikatorem użytkownika, który utworzył subskrypcję, lub jeśli został on później zmieniony przez inny identyfikator użytkownika, jest to identyfikator użytkownika ostatniej pomyślnej zmiany. Jeśli użyto parametru *SDAU* (i zezwolono na użycie alternatywnych identyfikatorów użytkowników dla tego użytkownika), alternatywny identyfikator użytkownika jest rejestrowany jako identyfikator użytkownika, który utworzył subskrypcję, a nie jako identyfikator użytkownika, w ramach którego dokonano subskrypcji.

Jeśli istnieje zgodna subskrypcja, która została utworzona bez opcji S0AUID , a identyfikator użytkownika subskrypcji różni się od identyfikatora użytkownika aplikacji żądającej uchwytu subskrypcji, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2434 .

Jeśli zgodna subskrypcja istnieje i jest obecnie używana przez inną aplikację, wywołanie kończy się niepowodzeniem z błędem RC2429 . Jeśli jest ona obecnie używana przez to samo połączenie, wywołanie nie kończy się niepowodzeniem i zwracany jest uchwyt do subskrypcji.

Jeśli subskrypcja o nazwie określonej w polu SubName nie jest poprawną subskrypcją, która może zostać wznowiona lub zmieniona w aplikacji, wywołanie kończy się niepowodzeniem z błędem RC2523 .

W poniższych tabelach przedstawiono atrybuty subskrypcji, które mogą być zmieniane przez S0ALT.

Tabela 728. Atrybuty w tabelach MQSD i MQSUB, które można zmieniać

Deskryptor typu danych lub wywołanie funkcji	Nazwa pola	Czy można zmienić ten atrybut za pomocą SOALT?	Kod przyczyny
Usługa MQSD	Opcje trwałości	Nie	RC2509
Usługa MQSD	Opcje miejsca docelowego	Tak	Brak
Usługa MQSD	Opcje rejestracji	Tak (patrz uwaga 1)	RC2515 w przypadku próby zmiany parametru SOGRP
Usługa MQSD	Opcje publikacji	Tak (patrz uwaga 2)	Brak
Usługa MQSD	Opcje znaku wieloznacznego	Nie	RC2510

Tabela 728. Atrybuty w tabelach MQSD i MQSUB, które można zmieniać (kontynuacja)			
Deskryptor typu danych lub wywołanie funkcji	**Nazwa pola**	**Czy można zmienić ten atrybut za pomocą SOALT?**	**Kod przyczyny**
Usługa MQSD	Inne opcje	Nie (patrz uwaga 3)	Brak
Usługa MQSD	ObjectName	Nie	RC2510
Usługa MQSD	SDAU	Nie (patrz uwaga 4)	Brak
Usługa MQSD	SDASI,	Nie (patrz uwaga 4)	Brak
Usługa MQSD	SDEXP,	Tak	Brak
Usługa MQSD	SDOSShortcut	Nie	RC2510
Usługa MQSD	SDSN	Nie (patrz uwaga 5)	Brak
Usługa MQSD	SSDSUD	Tak	Brak
Usługa MQSD	ID_DCID	Tak (patrz uwaga 6)	RC2515 w przypadku zgrupowanej subskrypcji
Usługa MQSD	SSDPRI	Tak	Brak
Usługa MQSD	SSDACC	Tak	Brak
Usługa MQSD	SDAID	Tak	Brak
Usługa MQSD	SDSL,	Nie	RC2512
MQSUB	Hobj	Tak (patrz uwaga 6)	RC2515 w przypadku zgrupowanej subskrypcji

Uwagi:

1. Nie można zmienić SOGRP .
2. Nie można zmienić parametru SONEWP , ponieważ nie jest on częścią subskrypcji.
3. Te opcje nie są częścią subskrypcji
4. Ten atrybut nie jest częścią subskrypcji
5. Ten atrybut jest tożsamością zmienianej subskrypcji
6. Możliwość zmiany, z wyjątkiem sytuacji, gdy część zgrupowanego elementu podrzędnego (SOGRP)

Opcje trwałości: Następujące opcje sterują trwałością subskrypcji. Można określić tylko jedną z tych opcji. Jeśli istniejąca subskrypcja jest zmieniana za pomocą opcji SOALT , nie można zmienić trwałości subskrypcji. W przypadku powrotu z wywołania MQSUB z użyciem SORESustawiana jest odpowiednia opcja trwałości.

SODUR
Zażądaj, aby subskrypcja tego tematu pozostała, dopóki nie zostanie jawnie usunięta za pomocą komendy MQCLOSE z opcją CORMSB . Jeśli ta subskrypcja nie zostanie jawnie usunięta, pozostanie ona nawet wtedy, gdy aplikacja nawiąże połączenie z menedżerem kolejek.

Jeśli żądanie trwałej subskrypcji zostanie wysłane do tematu, który jest zdefiniowany jako niezezwalający na trwałe subskrypcje, wywołanie zakończy się niepowodzeniem z błędem RC2436 .

SONDUR
Zażądaj, aby subskrypcja tego tematu została usunięta po zamknięciu połączenia aplikacji z menedżerem kolejek, jeśli nie została jeszcze jawnie usunięta. SONDUR jest przeciwieństwem opcji SODUR i jest definiowany w celu wspomagania dokumentacji programu. Jest to wartość domyślna, jeśli nie określono żadnej z nich.

Opcje miejsca docelowego: Następujące opcje sterują miejscem docelowym, do którego są wysyłane publikacje dla tematu, który został zasubskrybowany. W przypadku zmiany istniejącej subskrypcji przy użyciu opcji SOALT można zmienić miejsce docelowe używane na potrzeby publikacji dla subskrypcji. W przypadku powrotu z wywołania MQSUB z użyciem S0RESta opcja jest ustawiana w razie potrzeby.

SOMAN,

Zażądaj, aby miejsce docelowe, do którego są wysyłane publikacje, było zarządzane przez menedżer kolejek.

Uchwyt obiektu zwrócony w programie *HOBJ* reprezentuje kolejkę zarządzaną przez menedżer kolejek i jest przeznaczony do użycia z kolejnymi wywołaniami MQGET, MQCB, MQINQ lub MQCLOSE.

Nie można podać uchwytu obiektu zwróconego z poprzedniego wywołania MQSUB w parametrze **Hobj** , jeśli nie określono parametru S0MAN .

Opcje rejestracji: Następujące opcje sterują szczegółami rejestracji dokonanej w menedżerze kolejek dla tej subskrypcji. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji S0ALT można zmienić te opcje rejestracji. Przy powrocie z wywołania MQSUB z użyciem S0RES ustawiane są odpowiednie opcje rejestracji.

SOGRP

Ta subskrypcja jest pogrupowana z innymi subskrypcjami tego samego produktu *SDSL* przy użyciu tej samej kolejki i tego samego identyfikatora korelacji. Dzięki temu wszystkie publikacje do tematów, które spowodowałyby dostarczenie więcej niż jednego komunikatu publikacji do grupy subskrypcji, z powodu użycia nakładającego się zestawu łańcuchów tematów, tylko jeden komunikat zostanie dostarczony do kolejki. Jeśli ta opcja nie jest używana, każda unikalna subskrypcja (identyfikowana przez *SDSN*), która jest zgodna, jest dostarczana z kopią publikacji, co może oznaczać, że więcej niż jedna kopia publikacji może zostać umieszczona w kolejce współużytkowanej przez wiele subskrypcji.

Tylko najbardziej znacząca subskrypcja w grupie jest dostarczana wraz z kopią publikacji. Najbardziej znacząca subskrypcja jest oparta na pełnej nazwie tematu aż do miejsca, w którym znaleziono znak wieloznaczny. Jeśli w grupie używane są różne schematy znaków wieloznacznych, ważna jest tylko pozycja znaku wieloznacznego. Zaleca się, aby nie łączyć różnych schematów znaków wieloznacznych w obrębie grupy subskrypcji, które współużytkują tę samą kolejkę.

Podczas tworzenia nowej zgrupowanej subskrypcji musi ona nadal mieć unikalny identyfikator *SDSN*, ale jeśli jest zgodna z pełną nazwą tematu istniejącej subskrypcji w grupie, wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2514 .

Jeśli dla najbardziej istotnej subskrypcji w grupie określono również wartość S0NOLC , a jest to publikacja z tej samej aplikacji, do kolejki nie jest dostarczana żadna publikacja.

Podczas modyfikowania subskrypcji przy użyciu tej opcji nie można zmieniać pól, które implikują grupowanie, *Hobj* w wywołaniu MQSUB (reprezentującym nazwę kolejki i menedżera kolejek) oraz *SDCID* . Próba zmodyfikowania ich spowoduje, że wywołanie nie powiedzie się i zostanie wygenerowany błąd RC2515 .

Ta opcja musi być połączona z opcją S0SCID z wartością *SDCID* , która nie jest ustawiona na CIN0NEi nie może być łączona z wartością S0MAN.

SAUID

Jeśli określono wartość S0AUID , tożsamość subskrybenta nie jest ograniczona do pojedynczego identyfikatora użytkownika. Dzięki temu każdy użytkownik może zmienić lub wznowić subskrypcję, gdy ma odpowiednie uprawnienia. W danym momencie subskrypcja może być dostępna tylko dla jednego użytkownika. Próba wznowienia korzystania z subskrypcji obecnie używanej przez inną aplikację powoduje, że wywołanie kończy się niepowodzeniem z błędem RC2429 .

Aby dodać tę opcję do istniejącej subskrypcji, wywołanie MQSUB przy użyciu komendy S0ALTmusi pochodzić z tego samego identyfikatora użytkownika, co oryginalna subskrypcja.

Jeśli wywołanie MQSUB odwołuje się do istniejącej subskrypcji z ustawioną wartością SOAUID , a identyfikator użytkownika różni się od oryginalnej subskrypcji, wywołanie powiedzie się tylko wtedy, gdy nowy identyfikator użytkownika ma uprawnienie do subskrybowania tematu. Po pomyślnym zakończeniu działania przyszłe publikacje dla tego subskrybenta są umieszczane w kolejce subskrybenta z nowym identyfikatorem użytkownika ustawionym w komunikacie o publikacji.

Nie należy podawać jednocześnie parametrów SOAUID i SOFUID. Jeśli żadna z tych wartości nie zostanie podana, wartością domyślną jest SOFUID.

SOFUID

Jeśli określono wartość SOFUID , subskrypcja może zostać zmieniona lub wznowiona tylko przez ostatniego użytkownika, który ją zmodyfikuje. Jeśli subskrypcja nie została zmieniona, jest to identyfikator użytkownika, który ją utworzył.

Jeśli komenda MQSUB odwołuje się do istniejącej subskrypcji z ustawioną wartością SOAUID i modyfikuje subskrypcję za pomocą komendy SOALT w celu użycia identyfikatora SOFUID, identyfikator użytkownika subskrypcji jest teraz ustawiany na ten nowy identyfikator użytkownika. Wywołanie powiedzie się tylko wtedy, gdy nowy ID użytkownika ma uprawnienia do subskrybowania tematu.

Jeśli identyfikator użytkownika inny niż zarejestrowany jako właściciel subskrypcji próbuje wznowić lub zmienić subskrypcję SOFUID , wywołanie kończy się niepowodzeniem z błędem RC2434 . Identyfikator użytkownika będącego właścicielem subskrypcji można wyświetlić za pomocą komendy **DISPLAY SBSTATUS** .

Nie należy podawać jednocześnie parametrów SOAUID i SOFUID. Jeśli żadna z tych wartości nie zostanie podana, wartością domyślną jest SOFUID.

Opcje publikacji: Następujące opcje sterują sposobem wysyłania publikacji do tego subskrybenta. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji SOALT można zmienić te opcje publikacji.

SONOLC,
Informuje broker, że aplikacja nie chce wyświetlać żadnych własnych publikacji. Publikacje są uznawane za pochodzące z tej samej aplikacji, jeśli uchwyty połączenia są takie same. W przypadku powrotu z wywołania MQSUB z użyciem opcji SORES ta opcja jest ustawiana w razie potrzeby.

SONEWP,
Podczas tworzenia tej subskrypcji nie są wysyłane żadne zachowane publikacje, a tylko nowe. Ta opcja ma zastosowanie tylko wtedy, gdy określono opcję SOCRE . Wszelkie późniejsze zmiany w subskrypcji nie wpływają na przepływ publikacji, dlatego wszystkie publikacje, które zostały zachowane w temacie, zostały już wysłane do subskrybenta jako nowe publikacje.

Jeśli ta opcja zostanie podana bez opcji SOCRE , wywołanie zakończy się niepowodzeniem z błędem RC2046 . Po powrocie z wywołania MQSUB używającego opcji SORES ta opcja nie jest ustawiana, nawet jeśli subskrypcja została utworzona przy użyciu tej opcji.

Jeśli ta opcja nie jest używana, poprzednio zachowane komunikaty są wysyłane do podanej kolejki docelowej. Jeśli to działanie nie powiedzie się z powodu błędu (RC2525 lub RC2526), utworzenie subskrypcji nie powiedzie się.

Ta opcja nie jest poprawna w połączeniu z opcją SOPUBR.

SOPUBR
Ustawienie tej opcji wskazuje, że subskrybent żąda informacji dokładnie wtedy, gdy jest to wymagane. Menedżer kolejek nie wysyła niezamówionych komunikatów do subskrybenta. Zachowana publikacja (lub być może wiele publikacji, jeśli w temacie określono znak wieloznaczny) jest wysyłana do subskrybenta za każdym razem, gdy wywołanie MQSUBRQ jest wykonywane przy użyciu uchwytu Hsub z poprzedniego wywołania MQSUB. Przy użyciu tej opcji nie są wysyłane żadne publikacje w wyniku wywołania MQSUB. W przypadku powrotu z wywołania MQSUB z użyciem opcji SORES ta opcja jest ustawiana w razie potrzeby.

Ta opcja nie jest poprawna w połączeniu z opcją SONEWP.

Opcje znaków wieloznacznych: Następujące opcje sterują sposobem interpretowania znaków wieloznacznych w łańcuchu podanym w polu *SDOS* dokumentu MQSD. Można określić tylko jedną z tych opcji. W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji SOALT nie można zmieniać tych opcji ze znakami wieloznacznymi. W przypadku powrotu z wywołania MQSUB z użyciem SORES ustawiana jest odpowiednia opcja znaku wieloznacznego.

SOWCHR

Znaki wieloznaczne działają tylko na znakach w łańcuchu tematu. Pole SOWCHR traktuje ukośnik (/) jako kolejny znak bez specjalnego znaczenia.

Zachowanie zdefiniowane przez SOWCHR przedstawiono w poniższej tabeli:

Tabela 729. Interpretowanie znaków wieloznacznych

Znak specjalny	**zachowanie;**
*	Znak wieloznaczny, zero lub więcej znaków
?	Znak wieloznaczny, jeden znak
%	Znak zmiany znaczenia, który umożliwia użycie znaków '*', '?' lub '%' w łańcuchu i nie jest interpretowany jako znak specjalny, na przykład '% *', '%?' lub' %% '.

Na przykład publikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

jest zgodne z subskrybentami przy użyciu następujących tematów:

```
*
/*
/ level0/level1/level2/level3/*
/ level0/level1/*/level3/level4
/ level0/level1/le?el2/level3/level4
```

Uwaga: Użycie znaków wieloznacznych podczas używania komunikatów w formacie MQRFH1 do publikowania/subskrypcji jest dokładnie takie samo znaczenie, jak w przypadku komunikatów w formacie IBM MQ V6 i WebSphere MB V6 . Zaleca się, aby ta opcja nie była używana w przypadku nowo napisanych aplikacji i była używana tylko w przypadku aplikacji, które wcześniej były uruchamiane dla tej wersji i nie zostały zmienione w celu użycia domyślnego zachowania znaku wieloznacznego, zgodnie z opisem w sekcji SOWTOP.

SOWTOP

Znaki wieloznaczne działają tylko na elementach tematu w łańcuchu tematu. Jest to zachowanie domyślne, jeśli nie zostanie wybrana żadna opcja.

Zachowanie wymagane przez komendę SOWTOP przedstawiono w poniższej tabeli:

Tabela 730. Interpretowanie znaków wieloznacznych

Znak specjalny	**zachowanie;**
/	Separator poziomu tematu
#	Znak wieloznaczny: poziom wielu tematów
+	Znak wieloznaczny: poziom pojedynczego tematu

Uwaga:

Znaki '+' i '#' nie są traktowane jako znaki wieloznaczne, jeśli są mieszane z innymi znakami (w tym z samymi sobą) na poziomie tematu. W poniższym łańcuchu znaki '#' i '+' są traktowane jako zwykłe znaki.

```
level0/level1/#+/level3/level#
```

Na przykład publikowanie w następującym temacie:

```
/level0/level1/level2/level3/level4
```

jest zgodne z subskrybentami przy użyciu następujących tematów:

```
#
/#
/ level0/level1/level2/level3/#
/ level0/level1/+/level3/level4
```

Uwaga: Użycie znaków wieloznacznych podczas publikowania/subskrypcji ma znaczenie określone w sekcji WebSphere Message Broker 6 w przypadku komunikatów w formacie MQRFH2 .

Inne opcje: następujące opcje sterują sposobem wywoływania interfejsu API, a nie subskrypcją. W przypadku powrotu z wywołania MQSUB używającego S0RES te opcje nie ulegają zmianie.

SOALTU
Pole SDAU zawiera identyfikator użytkownika używany do sprawdzania poprawności tego wywołania MQSUB. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy dana jednostka SDAU ma uprawnienia do otwierania obiektu z określonymi opcjami dostępu, bez względu na to, czy identyfikator użytkownika, pod którym działa aplikacja, jest do tego uprawniony.

Identyfikator SOSCID
Subskrypcja ma używać identyfikatora korelacji podanego w polu *SDCID* . Jeśli ta opcja nie zostanie podana, identyfikator korelacji zostanie automatycznie utworzony przez menedżer kolejek w czasie subskrypcji i zwrócony do aplikacji w polu *SDCID* . Więcej informacji na ten temat zawiera sekcja SDCID (24-bajtowy łańcuch bitowy) SDCID .

SOSETI

Subskrypcja ma używać tokenu rozliczania i danych tożsamości aplikacji podanych w polach *SDACC* i *SDAID* .

Jeśli ta opcja jest określona, wykonywane jest takie samo sprawdzenie autoryzacji, jak w przypadku, gdy dostęp do kolejki docelowej był uzyskiwany za pomocą wywołania MQOPEN z parametrem 00SETI, z wyjątkiem sytuacji, gdy używana jest również opcja S0MAN . W takim przypadku w kolejce docelowej nie jest wykonywane sprawdzenie autoryzacji.

Jeśli ta opcja nie zostanie podana, publikacje wysyłane do tego subskrybenta będą mieć powiązane domyślne informacje o kontekście w następujący sposób:

Tabela 731. Domyślne informacje o kontekście dla publikacji wysyłanych do tego subskrybenta

Pole w strukturze MQMD	**Użyta wartość**
MDUID	Identyfikator użytkownika powiązany z subskrypcją w momencie jej wykonywania.
MDACC	Określana na podstawie środowiska, jeśli to możliwe; w przeciwnym razie ustawiana jest wartość ACNONE.
MDAID	Ustaw na wartości puste

Ta opcja jest poprawna tylko z opcjami SOCRE i SOALT. Jeśli ta opcja jest używana z opcjami SORES, pola *SDACC* i *SDAID* są ignorowane, więc ta opcja nie ma zastosowania.

Jeśli subskrypcja zostanie zmieniona bez użycia tej opcji, gdy wcześniej subskrypcja dostarczyła informacje o kontekście tożsamości, dla zmienionej subskrypcji zostaną wygenerowane domyślne informacje o kontekście.

Jeśli subskrypcja zezwalająca różnym identyfikatorom użytkowników na korzystanie z niej z opcją SOAUIDjest wznawiana przez inny ID użytkownika, dla nowego ID użytkownika będącego właścicielem subskrypcji generowany jest domyślny kontekst tożsamości, a wszystkie kolejne publikacje zawierające nowy kontekst tożsamości są dostarczane.

SFIQ
Wywołanie MQSUB kończy się niepowodzeniem, jeśli menedżer kolejek jest w stanie wyciszania. W systemie z/OSw przypadku aplikacji CICS lub IMS ta opcja wymusza także niepowodzenie wywołania MQSUB, jeśli połączenie jest w stanie wyciszania.

SDAU (12-bajtowy łańcuch znaków)

Jeśli zostanie podana wartość SOALTU, to pole zawiera alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji subskrypcji i danych wyjściowych do kolejki docelowej (określonej w parametrze **Hobj** wywołania MQSUB) zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona.

Jeśli operacja powiedzie się, identyfikator użytkownika określony w tym polu jest rejestrowany jako identyfikator użytkownika będącego właścicielem subskrypcji zamiast identyfikatora użytkownika, pod którym aplikacja jest obecnie uruchomiona.

Jeśli określono wartość SOALTU i pole to jest całkowicie puste, aż do pierwszego znaku o kodzie zero lub końca pola, subskrypcja może zakończyć się powodzeniem tylko wtedy, gdy autoryzacja użytkownika nie musi subskrybować tego tematu przy użyciu podanych opcji lub kolejki docelowej dla danych wyjściowych.

Jeśli wartość SOALTU nie jest określona, to pole jest ignorowane.

W przypadku powrotu z wywołania MQSUB używającego SORESto pole pozostaje niezmienione.

Jest to pole wejściowe. Długość tego pola jest określona przez LNUID. Wartością początkową tego pola jest 12 pustych znaków.

SDPRI (10-cyfrowa liczba całkowita ze znakiem)

Jest to wartość znajdująca się w polu *MQPRI* deskryptora komunikatu (MQMD) wszystkich komunikatów publikacji zgodnych z tą subskrypcją. Więcej informacji na temat pola *MQPRI* w strukturze MQMD zawiera sekcja MDPRI.

Wartość musi być większa lub równa zero; zero jest najniższym priorytetem. Można również użyć następujących wartości specjalnych:

PRQDEF,

Jeśli kolejka subskrypcji jest podana w polu Hobj w wywołaniu MQSUB i nie jest zarządzanym uchwytem, priorytet dla komunikatu jest pobierany z atrybutu **DefPriority** tej kolejki. Jeśli tak zidentyfikowana kolejka jest kolejką klastra lub w ścieżce rozstrzygania nazwy kolejki istnieje więcej niż jedna definicja, priorytet jest określany, gdy komunikat publikacji jest umieszczany w kolejce zgodnie z opisem dla MDPRI.

Jeśli wywołanie MQSUB używa zarządzanego uchwytu, priorytet komunikatu jest pobierany z atrybutu **DefPriority** kolejki modelowej powiązanej z subskrybowanym tematem.

PRPUB
Priorytet komunikatu jest priorytetem oryginalnej publikacji. Jest to wartość początkowa pola.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji SOALT można zmienić wartość parametru *MQPRI* dla wszystkich przyszłych komunikatów publikacji.

W przypadku powrotu z wywołania MQSUB z użyciem S0RESw tym polu jest ustawiony bieżący priorytet używany dla subskrypcji.

SDRO (MQCHARV)

SDRO jest długą nazwą obiektu po tym, jak menedżer kolejek rozstrzyga nazwę podaną w parametrze *SDON*.

Jeśli w polu *SDOS* zostanie podana długa nazwa obiektu i w polu *SDON*nie zostanie podana żadna wartość, wartość zwrócona w tym polu będzie taka sama, jak w polu *SDOS*.

Jeśli to pole jest pomijane (to znaczy SDRO.VSBufSize ma wartość zero), parametr *SDRO* nie jest zwracany, ale zwracana jest długość w parametrze SDRO.VSLength. Jeśli długość jest krótsza niż pełna nazwa *SDRO*, jest ona obcinana i zwraca tyle znaków po prawej stronie, ile może zmieścić się w podanej długości.

Jeśli parametr *SDRO* został określony niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2520 .

SDSID (4-bajtowy łańcuch znaków)

Jest to identyfikator struktury; wartość musi być następująca:

SDSIDV
Identyfikator struktury deskryptora subskrypcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest SDSIDV

SDSL (10-cyfrowa liczba całkowita ze znakiem)

Jest to poziom powiązany z subskrypcją. Publikacje są dostarczane do tej subskrypcji tylko wtedy, gdy znajduje się ona w zestawie subskrypcji o najwyższej wartości *SDSL* mniejszej lub równej PubLevel użytej w czasie publikacji.

Wartość musi być z zakresu od 0 do 9. Zero jest najniższym poziomem.

Wartością początkową tego pola jest 1.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji S0ALT nie można zmienić wartości parametru *SDSL* .

SDSN (MQCHARV)

SDSN określa nazwę subskrypcji.

To pole jest wymagane tylko wtedy, gdy w parametrze *SDOPT* określono opcję S0DUR , ale jeśli została ona podana, jest ona używana również przez menedżer kolejek dla parametru S0NDUR . Jeśli parametr *SDSN* zostanie określony, musi być unikalny w obrębie menedżera kolejek, ponieważ jest to pole używane do identyfikowania subskrypcji.

Maksymalna długość łańcucha *SDSN* wynosi 10240.

To pole służy dwóm celom. W przypadku subskrypcji SODUR jest to sposób identyfikowania subskrypcji, która ma zostać wznowiona po jej utworzeniu, jeśli uchwyt do subskrypcji został zamknięty (przy użyciu opcji C0KPSB) lub został odłączony od menedżera kolejek. Zidentyfikowanie subskrypcji, która ma zostać usunięta po jej utworzeniu, jest wykonywane za pomocą wywołania MQSUB z opcją S0RES . Pole SDSN jest również wyświetlane w widoku administrowania subskrypcji w polu *SDSN* w obszarze DISPLAY SBSTATUS.

Jeśli parametr *SDSN* jest określony niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV , jeśli przekracza maksymalną długość lub jeśli jest pominięty, gdy jest wymagany (czyli *SDSN*). *VCHRL* ma wartość zero) lub jeśli przekracza maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2440 .

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji S0ALT nie można zmienić nazwy subskrypcji, ponieważ jest to pole używane do identyfikowania subskrypcji. Nie jest on zmieniany w danych wyjściowych wywołania MQSUB z opcją S0RES .

SDSS (MQCHARV)

SDSS to łańcuch, który udostępnia kryteria wyboru używane podczas subskrybowania komunikatów z tematu.

To pole o zmiennej długości jest zwracane w danych wyjściowych wywołania MQSUB przy użyciu opcji S0RES , jeśli podano bufor i jeśli w polu VSBufSizeznajduje się także dodatnia długość buforu. Jeśli w wywołaniu nie podano buforu, w polu VSLength tabeli MQCHARV zwracana jest tylko długość łańcucha wyboru. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w podanym buforze są zwracane tylko bajty VSBufSize .

Jeśli parametr *SDSS* został określony niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2519 .

SDSUD (MQCHARV)

Dane podane w subskrypcji w tym polu są dołączane jako właściwość komunikatu mq.SubUserData każdej publikacji wysyłanej do tej subskrypcji.

Maksymalna długość łańcucha *SDSUD* wynosi 10240.

Jeśli parametr *SDSUD* jest określony niepoprawnie, zgodnie z opisem sposobu użycia struktury MQCHARV lub jeśli przekracza on maksymalną długość, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2431.

Jest to pole wejściowe. Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQCHARV.

W przypadku modyfikowania istniejącej subskrypcji przy użyciu opcji S0ALT można zmienić dane użytkownika subskrypcji.

To pole o zmiennej długości jest zwracane w danych wyjściowych wywołania MQSUB przy użyciu opcji S0RES , jeśli podano bufor i w pliku *VSBufLen*występuje dodatnia długość buforu. Jeśli w wywołaniu nie podano buforu, w polu *VCHRL* komendy MQCHARV zwracana jest tylko długość danych użytkownika subskrypcji. Jeśli podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w podanym buforze zwracane są tylko *VSBufLen* bajtów.

SDVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

SDVER1
Version-1 Struktura deskryptora subskrypcji.

Następująca stała określa numer wersji bieżącej:

SDVERC,
Bieżąca wersja struktury deskryptora subskrypcji.

Jest to zawsze pole wejściowe. Wartością początkową pola jest SDVER1.

Wartości początkowe

Tabela 732. Pola w MQSD

Nazwa pola	Nazwa stałej	Wartość stałej
SDSID	SDSIDV	'SD¬¬'
SDVER	SDVER1	1
SDOPT	SONDUR	0

Tabela 732. Pola w MQSD (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
SDON	Brak	Puste
SDAU	Brak	Puste
SDASI	SINON	Wartości null
SDEXP	EIULIM	-1
SDOS	Nazwy i wartości zdefiniowane dla MQCHARV	
SDSN	Nazwy i wartości zdefiniowane dla MQCHARV	
SDSUD	Nazwy i wartości zdefiniowane dla MQCHARV	
SDCID	KINON	Wartości null
SDPRI	PRQDEF,	-3
SDACC	ACNONE (brak)	Wartości null
SDAID	Brak	Puste
SDSL	Brak	1
SDRO	Nazwy i wartości zdefiniowane w tabeli MQCHARV	
Uwaga: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSD Structure
D*
D* Structure identifier
D  SDSID                  1      4
D* Structure version number
D  SDVER                  5      8I 0
D* Options associated with subscribing
D  SDOPT                  9     12I 0
D* Object name
D  SDON                  13     60
D* Alternate user identifier
D  SDAU                  61     72
D* Alternate security identifier
D  SDASI                 73    112
D* Expiry of Subscription
D  SDEXP                113    116I 0
D* Object Long name
D SDOSP                 117    132*
D SDOSO                 133    136I 0
D SDOSS                 137    140I 0
D SDOSL                 141    144I 0
D SDOSC                 145    148I 0
D* Subscription name
D SDSNP                 149    164*
D SDSNO                 165    168I 0
D SDSNS                 169    172I 0
D SDSNL                 173    176I 0
D SDSNC                 177    180I 0
D* Subscription User data
D SDSUDP                181    196*
```

```
D SDSUDO                197    200I 0
D SDSUDS                201    204I 0
D SDSUDL                205    208I 0
D SDSUDC                209    212I 0
D* Correlation Id related to this subscription
D SDCID                 213    236
D* Priority set in publications
D SDPRI                 237    240I 0
D* Accounting Token set in publications
D SDACC                 241    272
D* Appl Identity Data set in publications
D SDAID                 273    304
D* Message Selector
D SDSSP                 305    320*
D SDSSO                 321    324I 0
D SDSSS                 325    328I 0
D SDSSL                 329    332I 0
D SDSSC                 333    336
D* Subscription level
D SDSL                  337    340  0
D* Resolved Long object name
D SDROP                 341    356*
D SDROO                 357    360I 0
D SDROS                 361    364I 0
D SDROL                 365    368I 0
D SDROC                 369    372I 0
```

IBM i MQSMPO (ustawianie opcji właściwości komunikatu) w systemie IBM i

Struktura **MQSMPO** umożliwia aplikacjom określanie opcji sterujących ustawianiem właściwości komunikatów.

Przegląd

Cel: Struktura jest parametrem wejściowym wywołania **MQSETMP** .

Zestaw znaków i kodowanie: dane w pliku **MQSMPO** muszą znajdować się w zestawie znaków aplikacji i kodowaniu aplikacji (ENNAT).

- "Pola" na stronie 1265
- "Wartości początkowe" na stronie 1266
- "Deklaracja RPG" na stronie 1267

Pola

Struktura MQSMPO zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

SPOPT (10-cyfrowa liczba całkowita ze znakiem)

Opcje położenia: Następujące opcje odnoszą się do względnego położenia właściwości w porównaniu z kursorem właściwości:

SPSETF,
Ustawia wartość pierwszej właściwości, która jest zgodna z podaną nazwą, lub, jeśli nie istnieje, dodaje nową właściwość po wszystkich innych właściwościach ze zgodną hierarchią.

SPSETC,
Ustawia wartość właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości jest tą, do której ostatnio wysłano zapytanie przy użyciu opcji IPINQF lub IPINQN.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany lub gdy uchwyt komunikatu jest określony w polu *HMSG* struktury MQGMO w wywołaniu MQGET lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze określony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2471.

SPSETA
Ustawia nową właściwość po właściwości wskazywanej przez kursor właściwości. Właściwość wskazywana przez kursor właściwości jest tą, do której ostatnio wysłano zapytanie przy użyciu opcji IPINQF lub IPINQO.

Kursor właściwości jest resetowany, gdy uchwyt komunikatu jest ponownie wykorzystywany lub gdy uchwyt komunikatu jest określony w polu *HMSG* struktury MQGMO w wywołaniu MQGET lub MQPMO w wywołaniu MQPUT.

Jeśli ta opcja jest używana, gdy kursor właściwości nie został jeszcze określony lub jeśli właściwość wskazywana przez kursor właściwości została usunięta, wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2471.

Jeśli żadna z opisanych opcji nie jest potrzebna, należy użyć następującej opcji:

SPNONE.
Nie określono żadnych opcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest SPSETF.

SPSID (10-cyfrowa liczba całkowita ze znakiem)

Jest to identyfikator struktury; wartość musi być następująca:

SPSIDV

Identyfikator struktury opcji ustawiania właściwości komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **SPSIDV**.

SPVAKCSI (10-cyfrowa liczba całkowita ze znakiem)

Zestaw znaków wartości właściwości, który ma zostać ustawiony, jeśli wartość jest łańcuchem znaków.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **CSAPL**.

SPVALENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie wartości właściwości, która ma zostać ustawiona, jeśli wartość jest wartością liczbową.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **ENNAT**.

SPVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

SPVER1
Version-1 służy do ustawiania struktury opcji właściwości komunikatu.

Następująca stała określa numer wersji bieżącej:

SPVERC
Bieżąca wersja struktury ustawionych opcji właściwości komunikatu.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest **SPVER1**.

Wartości początkowe

Tabela 733. Pola w MQSMPO

Nazwa pola	Nazwa stałej	Wartość stałej
SPSID	SPSIDV	'SMPO'

Tabela 733. Pola w MQSMPO (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
SPVER	SPVER1	1
SPOPT	SPNONE.	0
SPVALENC	ENNAT,	Zależy od środowiska
SPVALCSI	CSAPL	-3

Deklaracja RPG

```
D* MQSMPO Structure
D*
D*
D* Structure identifier
D  SPSID                  1      4    INZ('SMPO')
D*
D* Structure version number
D  SPVER                  5      8I 0 INZ(1)
D*
** Options that control the action of
D* MQSETMP
D  SPOPT                  9     12I 0 INZ(0)
D*
D* Encoding of Value
D  SPVALENC              13     16I 0 INZ(273)
D*
D* Character set identifier of Value
D  SPVALCSI              17     20I 0 INZ(-3)
```

IBM i MQSRO (opcje żądań subskrypcji) w systemie IBM i

Struktura MQSRO umożliwia aplikacji określenie opcji sterujących sposobem wysyłania żądań subskrypcji.

Przegląd

Cel: Struktura jest parametrem wejścia/wyjścia wywołania MQSUBRQ.

Wersja: Bieżąca wersja MQSRO to SRVER1.

- "Pola" na stronie 1267
- "Wartości początkowe" na stronie 1268
- "Deklaracja RPG" na stronie 1268

Pola

Struktura MQSRO zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

SRNMP (10-cyfrowa liczba całkowita ze znakiem)

Jest to pole wyjściowe zwracane do aplikacji w celu wskazania liczby publikacji wysłanych do kolejki subskrypcji w wyniku tego wywołania. Mimo że ta liczba publikacji została wysłana w wyniku tego wywołania, nie ma gwarancji, że ta liczba komunikatów będzie dostępna dla aplikacji, zwłaszcza jeśli są to komunikaty nietrwałe.

Jeśli subskrybowany temat zawiera znak wieloznaczny, może istnieć więcej niż jedna publikacja. Jeśli w łańcuchu tematu nie ma znaków wieloznacznych podczas tworzenia subskrypcji reprezentowanej przez *HSUB* , w wyniku tego wywołania zostanie wysłana co najwyżej jedna publikacja.

SROPT (10-cyfrowa liczba całkowita ze znakiem)

Należy podać jedną z następujących opcji. Można podać tylko jedną opcję.

Inne opcje: Poniższa opcja steruje tym, co się dzieje, gdy menedżer kolejek jest wyciszany:

Kolejka SRFIQ

Wywołanie MQSUBRQ nie powiedzie się, jeśli menedżer kolejek jest w stanie wyciszania.

Opcja domyślna: Jeśli opisana wcześniej opcja nie jest wymagana, należy użyć następującej opcji:

BRAK SRNONE

Wartość ta wskazuje, że nie określono innych opcji. Wszystkie opcje przyjmują wówczas wartości domyślne.

SRNONE pomaga w dokumentacji programu. Chociaż ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ponieważ jej wartość wynosi zero, nie można jej wykryć.

SRSID (4-bajtowy łańcuch znaków)

Jest to identyfikator struktury; wartość musi być następująca:

SRSIDV
Identyfikator struktury SROPT żądania subskrypcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest SRSIDV.

SRVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury; wartość musi być następująca:

SRVER1
Version-1 Struktura opcji żądania subskrypcji.

Następująca stała określa numer wersji bieżącej:

SWERC
Bieżąca wersja struktury opcji żądań subskrypcji.

Jest to zawsze pole wejściowe. Wartością początkową tego pola jest SRVER1.

Wartości początkowe

Tabela 734. Pola w MQSRO

Nazwa pola	Nazwa stałej	Wartość stałej
SRSID	SRSIDV	'SR0¬'
SRVER	SRVER1	1
SROPT	BRAK SRNONE	0
SRNMP	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. Wartość Null lub odstępy oznaczają łańcuch pusty w języku C i znaki puste w innych językach programowania.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSRO Structure
D*
D* Structure identifier
D  SRSID                  1      4
D* Structure version number
D  SRVER                  5      8I 0
D* Options that control the action of MQSUBRQ
```

```
D  SROPT                  9     12I 0
D* Number of publications sent
D  SRNMP                 13     16I 0
```

IBM i MQSTS (struktura raportowania statusu) w systemie IBM i

Struktura MQSTS opisuje dane w strukturze statusu zwróconej przez komendę MQSTAT.

Przegląd

Zestaw znaków i kodowanie: dane znakowe w MQSTS znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut *CodedCharSetId* menedżera kolejek. Dane liczbowe w usłudze MQSTS są kodowane na komputerze rodzimym. Jest to dane podawane przez komendę *ENNAT*.

Składnia: Komenda MQSTAT służy do pobierania informacji o statusie. Te informacje są zwracane w strukturze MQSTS. Więcej informacji na temat komendy MQSTAT zawiera sekcja "MQSTAT (Odtworzenie informacji o statusie) w systemie IBM i" na stronie 1401.

- "Pola" na stronie 1269
- "Wartości początkowe" na stronie 1272
- "Deklaracja RPG" na stronie 1273

Pola

Struktura MQSTS zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

STSCC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kod zakończenia wynikający z pierwszego błędu zgłoszonego w strukturze MQSTS.

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest CCOK.

STSFC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba asynchronicznych wywołań put, które zakończyły się niepowodzeniem.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0.

STSOBJN (48-bajtowy łańcuch znaków)

Jest to nazwa lokalna obiektu, którego dotyczy pierwsze niepowodzenie.

Jest to pole wyjściowe. Wartością początkową tego pola jest 48 znaków odstępu.

STSOQMGR (48-bajtowy łańcuch znaków)

Jest to nazwa menedżera kolejek, w którym zdefiniowano obiekt *STSOBJN* . Nazwa, która jest całkowicie pusta aż do pierwszego znaku o kodzie zero lub końca pola, oznacza menedżera kolejek, z którym połączona jest aplikacja (menedżer kolejek lokalnych).

Jest to pole wyjściowe. Wartością początkową tego pola jest 48 znaków odstępu.

STSOO (10-cyfrowa liczba całkowita ze znakiem)

STS00 używany do otwierania zgłaszanego obiektu. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

Wartość parametru STS00 zależy od wartości parametru MQSTAT **STYPE** .

STATAPTa

Zero.

STATYCZNY

Zero.

STATRER,

Wartość STS00 używana w momencie wystąpienia awarii. Przyczyna niepowodzenia jest zgłaszana w polach *STSCC* i *STSRC* w strukturze MQSTS .

STS00 jest zmienną wyjściową. Wartością początkową jest zero.

STSOS (MQCHARV)

Długa nazwa obiektu, dla którego zgłoszono błąd. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

STS0S jest polem MQCHARV o maksymalnej długości 10240. Opis sposobu użycia struktury MQCHARV zawiera sekcja MQCHARV .

Interpretacja parametru STS0S zależy od wartości parametru MQSTAT **STYPE** .

STATAPTa

Jest to długa nazwa obiektu kolejki lub tematu używanego w operacji MQPUT , która się nie powiodła.

STATYCZNY

Łańcuch o zerowej długości

STATRER,

Jest to długa nazwa obiektu, który spowodował niepowodzenie ponownego połączenia.

STS0S jest zmienną wyjściową. Jego wartością początkową jest łańcuch o zerowej długości.

STSOT (10-cyfrowa liczba całkowita ze znakiem)

Typ obiektu, którego nazwa znajduje się w pliku *ObjectName*. Dozwolone są następujące wartości:

OTALSQ
Kolejka aliasowa.

OTLOCQ (kolejka)
Kolejka lokalna.

Kolejka OTMODQ
Kolejka modelowa.

OTQ
do kolejki błędów.

OTREMQ
Kolejka zdalna.

OTTOP
.

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest OTQ.

STSRC (10-cyfrowa liczba całkowita ze znakiem)

Jest to kod przyczyny wynikający z pierwszego błędu zgłoszonego w strukturze MQSTS

Jest to zawsze pole wyjściowe. Wartością początkową tego pola jest RCNONE.

STSROBJN (48-bajtowy łańcuch znaków)

Jest to nazwa kolejki docelowej o nazwie określonej w polu *STSOBJN* po przetłumaczeniu nazwy przez lokalny menedżer kolejek. Zwracana nazwa jest nazwą kolejki, która istnieje w menedżerze kolejek identyfikowanym przez *STSRQMGR*.

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą do przeglądania, wprowadzania lub wyprowadzania (lub dowolnej kombinacji). Jeśli otwierany jest dowolny z następujących obiektów, parametr *STSROBJN* jest ustawiany na wartość pustą:

- Temat

- Kolejka, ale nie otwarta do przeglądania, wejścia lub wyjścia

Jest to pole wyjściowe. Wartością początkową tego pola jest 48 znaków odstępu.

STSRQMGR (48-bajtowy łańcuch znaków)

Jest to nazwa docelowego menedżera kolejek po przetłumaczeniu nazwy przez lokalny menedżer kolejek. Zwracana nazwa to nazwa menedżera kolejek, który jest właścicielem kolejki identyfikowanej przez *STSROBJN*. *STSRQMGR* może być nazwą lokalnego menedżera kolejek.

Jeśli *STSROBJN* jest kolejką współużytkowaną, której właścicielem jest grupa współużytkowania kolejek, do której należy lokalny menedżer kolejek, *STSRQMGR* jest nazwą grupy współużytkowania kolejek. Jeśli właścicielem kolejki jest inna grupa współużytkowania kolejek, parametr *STSROBJN* może być nazwą grupy współużytkowania kolejek lub nazwą menedżera kolejek, który jest elementem grupy współużytkowania kolejek (rodzaj zwracanej wartości jest określany przez definicje kolejek istniejące w lokalnym menedżerze kolejek).

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką otwartą do przeglądania, wprowadzania lub wyprowadzania (lub dowolnej kombinacji). Jeśli otwierany jest dowolny z następujących obiektów, parametr *STSRQMGR* jest ustawiany na wartość pustą:

- Temat
- Kolejka, ale nie otwarta do przeglądania, wejścia lub wyjścia
- Kolejka klastra z określoną wartością OOBNDN (lub z wartością OOBNDQ, jeśli atrybut kolejki **DefBind** ma wartość OOBNDN)

Jest to pole wyjściowe. Wartością początkową tego pola jest 48 znaków odstępu.

STSSC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba asynchronicznych wywołań umieszczania, które zakończyły się powodzeniem.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0.

STSSID (4-bajtowy łańcuch znaków)

Jest to identyfikator struktury. Wartość musi być następująca:

Identyfikator STSSID
Identyfikator struktury raportowania statusu.

Wartością początkową tego pola jest STSSID.

STSSO (10-cyfrowa liczba całkowita ze znakiem)

STSS0 używany do otwierania subskrypcji, która się nie powiodła. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

Interpretacja parametru STSS0 zależy od wartości parametru MQSTAT **STYPE** .

STATAPTa

Zero.

STATYCZNY

Zero.

STATRER,

Wartość STSS0 używana w momencie wystąpienia awarii. Przyczyna niepowodzenia jest zgłaszana w polach *STSCC* i *STSRC* w strukturze MQSTS . Jeśli niepowodzenie nie jest związane z subskrybowaniem tematu, zwracana jest wartość zero.

STSS0 jest zmienną wyjściową. Wartością początkową jest zero.

STSSUN (MQCHARV)

Nazwa subskrypcji zakończonej niepowodzeniem. Występuje tylko w wersji 2 systemu MQSTS lub nowszej.

STSSUN to pole MQCHARV o maksymalnej długości 10240. Opis sposobu użycia struktury MQCHARV zawiera sekcja MQCHARV .

Interpretacja parametru STSSUN zależy od wartości parametru MQSTAT **STYPE** .

STATAPTa

Łańcuch o zerowej długości.

STATYCZNY

Łańcuch o zerowej długości.

STATRER,

Nazwa subskrypcji, która spowodowała niepowodzenie ponownego połączenia. Jeśli nazwa subskrypcji nie jest dostępna lub niepowodzenie nie jest związane z subskrypcją, jest to łańcuch o zerowej długości.

STSSUN jest zmienną wyjściową. Jego wartością początkową jest łańcuch o zerowej długości.

STSVER (10-cyfrowa liczba całkowita ze znakiem)

Jest to numer wersji struktury. Wartość musi być następująca:

STSVR1
Numer wersji struktury raportowania statusu.

Następująca stała określa numer wersji bieżącej:

SSVRC
Bieżąca wersja struktury raportowania statusu.

Wartością początkową tego pola jest STSVR1.

STSWC (10-cyfrowa liczba całkowita ze znakiem)

Jest to liczba asynchronicznych wywołań put zakończonych z ostrzeżeniem.

Jest to pole wyjściowe. Wartością początkową tego pola jest 0.

Wartości początkowe

Tabela 735. Pola w MQSTS

Nazwa pola	Nazwa stałej	Wartość stałej
STSSID	IDENTYFIKATOR SID	
STSVER	SSVRC	STSVR1
STSCC	CKOK	0
STSRC	BRAK RCNONE	0
STSSC	Brak	0
STSWC	Brak	0
STSFC	Brak	0
STSOT	Brak	0
STSOBJN	Brak	Puste
STSOQMGR	Brak	Puste
STSROBJN	Brak	Puste
STSRQMGR	Brak	Puste
STSOS	Nazwy i wartości zdefiniowane dla MQCHARV	

Tabela 735. Pola w MQSTS (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
STSSUN	Nazwy i wartości zdefiniowane dla MQCHARV	
STS00	Brak	0
STSS0	Brak	0

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQSTS Structure
D*
D* Structure identifier
D  STSSID                 1      4
D* Structure version number
D  STSVER                 5      8I 0
D* Completion code
D  STSCC                  9     12I 0
D* Reason code
D  STSRC                 13     16I 0
D* Success count
D  STSSC                 17     20I 0
D* Warning count
D  STSWC                 21     24I 0
D* Failure count
D  STSFC                 25     28I 0
D* Object type
D  STSOT                 29     32I 0
D* Object name
D  STSOBJN               33     80
D* Object queue manager
D  STSOQMGR              81     128
D* Resolved object name
D  STSROBJN             129     176
D* Resolved object queue manager name
D  STSRQMGR             177     224
D* Ver:1 **
D* Failing object long name
D* Address of variable length string
D  STSOSCHRP            225    240*
D* Offset of variable length string
D  STSOSCHRO            241    244I 0
D* Size of buffer
D  STSOSVSBS            245    248I 0
D* Length of variable length string
D  STSOSCHRL            249    252I 0
D* CCSID of variable length string
D  STSOSCHRC            253    256I 0
D* Failing subscription name
D* Address of variable length string
D  STSSUNCHRP           257    272*
D* Offset of variable length string
D  STSSUNCHRO           273    276I 0
D* Size of buffer
D  STSSUNVSBS           277    280I 0
D* Length of variable length string
D  STSSUNCHRL           281    284I 0
D* CCSID of variable length string
D  STSSUNCHRC           285    288I 0
D* Failing open options
D  STSOO                289    292I 0
D* Failing subscription options
D  STSSO                293    296I 0
D* Ver:2 **
```

MQTM-komunikat wyzwalacza

Struktura MQTM opisuje dane w komunikacie wyzwalacza, który jest wysyłany przez menedżer kolejek do aplikacji monitorującej wyzwalacz, gdy dla kolejki wystąpi zdarzenie wyzwalacza.

Przegląd

Cel: ta struktura jest częścią interfejsu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska IBM MQ .

Nazwa formatu: FMTM.

Zestaw znaków i kodowanie: dane znakowe w produkcie MQTM znajdują się w zestawie znaków menedżera kolejek, który generuje produkt MQTM. Dane liczbowe w produkcie MQTM są kodowane na komputerze menedżera kolejek, który generuje produkt MQTM.

Zestaw znaków i kodowanie produktu MQTM są określone przez pola *MDCSI* i *MDENC* w następujących polach:

- MQMD (jeśli struktura MQTM znajduje się na początku danych komunikatu), lub
- Struktura nagłówka poprzedzająca strukturę MQTM (wszystkie inne przypadki).

Użycie: Aplikacja monitora wyzwalacza może wymagać przekazania niektórych lub wszystkich informacji zawartych w komunikacie wyzwalacza do aplikacji uruchamianej przez aplikację monitora wyzwalacza. Informacje, które mogą być wymagane przez uruchomioną aplikację, obejmują *TMQN*, *TMTD*i *TMUD*. Aplikacja monitora wyzwalacza może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2 , w zależności od tego, co jest dozwolone w środowisku i wygodne dla uruchomionej aplikacji. Więcej informacji na temat komendy MQTMC2zawiera sekcja "MQTMC2 (format 2-znakowy komunikatu wyzwalacza) w systemie IBM i" na stronie 1278.

- W systemie IBM iaplikacja monitora wyzwalacza dostarczana z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.

Informacje na temat wyzwalaczy zawiera sekcja Wymagania wstępne do wyzwalania.

MQMD dla komunikatu wyzwalacza

Tabela 736. Ustawienia pól w strukturze MQMD komunikatu wyzwalacza wygenerowanego przez menedżer kolejek

Pole w strukturze MQMD	Użyta wartość
MDSID	MDSIDV
MDVER	MDVER1
MDREP	BRAK
MDMT	MTGRM
MDEXP	EIULIM
MDFB	FBNONE
MDENC	ENNAT,
MDCSI	Atrybut **CodedCharSetId** menedżera kolejek
MDFMT	FMTM,
MDPRI	Atrybut **DefPriority** kolejki inicjującej
MDPER	PENPER
MDMID	Wartość unikalna
MDCID	KINON

Tabela 736. Ustawienia pól w strukturze MQMD komunikatu wyzwalacza wygenerowanego przez menedżer kolejek (kontynuacja)

Pole w strukturze MQMD	Użyta wartość
MDBOC	0
MDRQ	Puste
MDRM	Nazwa menedżera kolejek.
MDUID	Puste
MDACC	ACNONE (brak)
MDAID	Puste
MDPAT	ATQM lub odpowiednio dla agenta kanału komunikatów
MDPAN	Pierwsze 28 bajtów nazwy menedżera kolejek
MDPD	Data wysłania komunikatu wyzwalacza
MDPT	Czas wysłania komunikatu wyzwalacza
MDAOD	Puste

Zaleca się, aby aplikacja generująca komunikat wyzwalacza ustawiła podobne wartości, z wyjątkiem następujących:

- Pole *MDPRI* można ustawić na wartość PRQDEF (menedżer kolejek zmieni ten priorytet na domyślny priorytet kolejki inicjującej po umieszczeniu komunikatu).
- Pole *MDRM* można ustawić na wartość pustą (menedżer kolejek zmieni tę wartość na nazwę lokalnego menedżera kolejek podczas umieszczania komunikatu).
- Pola kontekstu powinny być ustawione odpowiednio dla aplikacji.

Pola

Struktura MQTM zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

TMAI (256-bajtowy łańcuch znaków)

Identyfikator aplikacji.

Jest to łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona i używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **ApplId** obiektu procesu identyfikowanego przez pole *TMPN* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów w systemie IBM i” na stronie 1441 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

Znaczenie parametru *TMAI* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniony przez IBM MQ wymaga, aby *TMAI* była nazwą programu wykonywalnego.

Długość tego pola jest określona przez LNPROA. Wartością początkową tego pola jest 256 znaków odstępu.

TMAT (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji.

Identyfikuje on rodzaj programu, który ma zostać uruchomiony i jest używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **ApplType** obiektu procesu identyfikowanego przez pole *TMPN* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów w systemie IBM i” na stronie 1441 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

TMAT może mieć jedną z następujących wartości standardowych. Typy zdefiniowane przez użytkownika mogą być również używane, ale powinny być ograniczone do wartości z zakresu od ATUFST do ATULST:

oCICS
CICS .

ATVSE,
CICS/VSE .

AT400
Aplikacja IBM i .

ATUFST
Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

ATULST
Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Wartością początkową tego pola jest 0.

TMED (128-bajtowy łańcuch znaków)

Dane środowiska.

Jest to łańcuch znaków, który zawiera informacje związane ze środowiskiem dotyczące aplikacji, która ma zostać uruchomiona, i jest używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **EnvData** obiektu procesu identyfikowanego przez pole *TMPN* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów w systemie IBM i” na stronie 1441 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

Długość tego pola jest określona przez LNPROE. Wartością początkową tego pola jest 128 znaków odstępu.

TMPN (48-bajtowy łańcuch znaków)

Nazwa obiektu procesu.

Jest to nazwa obiektu procesu menedżera kolejek określona dla wyzwalanej kolejki i może być używana przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **ProcessName** kolejki identyfikowanej przez pole *TMQN* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty kolejek” na stronie 1410 .

Nazwy krótsze niż zdefiniowana długość pola są zawsze dopełniane do prawej strony odstępami; nie są przedwcześnie kończone znakiem o kodzie zero.

Długość tego pola jest określona przez LNPRON. Wartością początkową tego pola jest 48 znaków odstępu.

TMQN (48-bajtowy łańcuch znaków)

Nazwa wyzwalanej kolejki.

Jest to nazwa kolejki, dla której wystąpiło zdarzenie wyzwalające i jest używana przez aplikację uruchomioną przez aplikację monitorującą wyzwalacz. Menedżer kolejek inicjuje to pole wartością atrybutu **QName** wyzwalanej kolejki. Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty kolejek” na stronie 1410 .

Nazwy, które są krótsze niż zdefiniowana długość pola, są dopełniane do prawej strony odstępami; nie są przedwcześnie kończone znakiem o kodzie zero.

Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

TMSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

TMSIDV
Identyfikator struktury komunikatu wyzwalacza.

Wartością początkową tego pola jest TMSIDV.

TMTD (64-bitowy łańcuch znaków)

Dane wyzwalacza.

Są to dane w formacie swobodnym używane przez aplikację monitora wyzwalacza, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **TriggerData** kolejki identyfikowanej przez pole *TMQN* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty kolejek” na stronie 1410 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

Długość tego pola jest określona przez LNTRGD. Wartością początkową tego pola jest 64 znaki odstępu.

TMUD (128-bajtowy łańcuch znaków)

Dane użytkownika.

Jest to łańcuch znaków, który zawiera informacje o użytkowniku dotyczące uruchamianej aplikacji i jest używany przez aplikację monitorującą wyzwalacz, która odbiera komunikat wyzwalacza. Menedżer kolejek inicjuje to pole wartością atrybutu **UserData** obiektu procesu identyfikowanego przez pole *TMPN* . Szczegółowe informacje na temat tego atrybutu zawiera sekcja “Atrybuty definicji procesów w systemie IBM i” na stronie 1441 . Treść tych danych nie ma znaczenia dla menedżera kolejek.

Długość tego pola jest określona przez LNPROU. Wartością początkową tego pola jest 128 znaków odstępu.

TMVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

TMVER1
Numer wersji struktury komunikatu wyzwalacza.

Następująca stała określa numer wersji bieżącej:

TMVERC
Bieżąca wersja struktury komunikatu wyzwalacza.

Wartością początkową tego pola jest TMVER1.

Wartości początkowe

Tabela 737. Pola w MQTM

Nazwa pola	Nazwa stałej	Wartość stałej
TMSID	TMSIDV	'TM¬¬'
TMVER	TMVER1	1
TMQN	Brak	Puste
TMPN	Brak	Puste
TMTD	Brak	Puste
TMAT	Brak	0
TMAI	Brak	Puste
TMED	Brak	Puste

Tabela 737. Pola w MQTM (kontynuacja)		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
TMUD	Brak	Puste
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQTM Structure
D*
D* Structure identifier
D  TMSID                  1      4    INZ('TM  ')
D* Structure version number
D  TMVER                  5      8I 0 INZ(1)
D* Name of triggered queue
D  TMQN                   9     56    INZ
D* Name of process object
D  TMPN                  57    104    INZ
D* Trigger data
D  TMTD                 105    168    INZ
D* Application type
D  TMAT                 169    172I 0 INZ(0)
D* Application identifier
D  TMAI                 173    428    INZ
D* Environment data
D  TMED                 429    556    INZ
D* User data
D  TMUD                 557    684    INZ
```

IBM i MQTMC2 (format 2-znakowy komunikatu wyzwalacza) w systemie IBM i

Gdy aplikacja monitora wyzwalacza pobiera komunikat wyzwalacza (MQTM) z kolejki inicjującej, może być konieczne przekazanie przez monitor wyzwalacza niektórych lub wszystkich informacji zawartych w komunikacie wyzwalacza do aplikacji uruchamianej przez monitor wyzwalacza.

Przegląd

Cel: informacje, które mogą być wymagane przez uruchomioną aplikację, obejmują *TC2QN*, *TC2TD*i *TC2UD*. Aplikacja monitora wyzwalacza może przekazać strukturę MQTM bezpośrednio do uruchomionej aplikacji lub przekazać strukturę MQTMC2 , w zależności od tego, co jest dozwolone w środowisku i wygodne dla uruchomionej aplikacji.

Ta struktura jest częścią interfejsu IBM MQ Trigger Monitor Interface (TMI), który jest jednym z interfejsów środowiska IBM MQ .

Zestaw znaków i kodowanie: dane znakowe w programie MQTMC2 znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek systemu **CodedCharSetId** .

Użycie: Struktura MQTMC2 jest podobna do formatu struktury MQTM. Różnica polega na tym, że pola nieznakowe w programie MQTM są zmieniane w programie MQTMC2 na pola znakowe o tej samej długości, a nazwa menedżera kolejek jest dodawana na końcu struktury.

- W systemie IBM iaplikacja monitora wyzwalacza dostarczana z produktem IBM MQ przekazuje strukturę MQTMC2 do uruchomionej aplikacji.
- “Pola” na stronie 1279
- “Wartości początkowe” na stronie 1280

- “Deklaracja RPG” na stronie 1280

Pola

Struktura MQTMC2 zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

TC2AI (256-bajtowy łańcuch znaków)

Identyfikator aplikacji.

Patrz pole *TMAI* w strukturze MQTM.

TC2AT (4-bajtowy łańcuch znaków)

Typ aplikacji.

To pole zawsze zawiera odstępy, niezależnie od wartości w polu *TMAT* w strukturze MQTM oryginalnego komunikatu wyzwalacza.

TC2ED (128-bajtowy łańcuch znaków)

Dane środowiska.

Patrz pole *TMED* w strukturze MQTM.

TC2PN (48-bajtowy łańcuch znaków)

Nazwa obiektu procesu.

Patrz pole *TMPN* w strukturze MQTM.

TC2QMN (48-bajtowy łańcuch znaków)

Nazwa menedżera kolejek.

Jest to nazwa menedżera kolejek, w którym wystąpiło zdarzenie wyzwalające.

TC2QN (48-bajtowy łańcuch znaków)

Nazwa wyzwalanej kolejki.

Patrz pole *TMQN* w strukturze MQTM.

TC2SID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

TCSIDV
Identyfikator struktury komunikatu wyzwalacza (format znakowy).

TC2TD (64-bitowy łańcuch znaków)

Dane wyzwalacza.

Patrz pole *TMTD* w strukturze MQTM.

TC2UD (128-bajtowy łańcuch znaków)

Dane użytkownika.

Patrz pole *TMUD* w strukturze MQTM.

TC2VER (4-bajtowy łańcuch znaków)

Numer wersji struktury.

Wartość musi być następująca:

TCVER2
Struktura komunikatu wyzwalacza (format znakowy) w wersji 2.

Następująca stała określa numer wersji bieżącej:

TCVERC
Bieżąca wersja struktury komunikatu wyzwalacza (format znakowy).

Wartości początkowe

Tabela 738. Pola w tabeli MQTMC2		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
TC2SID	TCSIDV	'TMC¬'
TC2VER	TCVER2	'¬¬¬2'
TC2QN	Brak	Puste
TC2PN	Brak	Puste
TC2TD	Brak	Puste
TC2AT	Brak	Puste
TC2AI	Brak	Puste
TC2ED	Brak	Puste
TC2UD	Brak	Puste
TC2QMN	Brak	Puste
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQTMC2 Structure
D*
D* Structure identifier
D  TC2SID                1      4
D* Structure version number
D  TC2VER                5      8
D* Name of triggered queue
D  TC2QN                 9     56
D* Name of process object
D  TC2PN                57    104
D* Trigger data
D  TC2TD               105    168
D* Application type
D  TC2AT               169    172
D* Application identifier
D  TC2AI               173    428
D* Environment data
D  TC2ED               429    556
D* User data
D  TC2UD               557    684
D* Queue manager name
D  TC2QMN              685    732
```

IBM i MQWIH (nagłówek informacji o pracy) w systemie IBM i

Struktura MQWIH opisuje informacje, które muszą być obecne na początku komunikatu, który ma być obsługiwany przez menedżer obciążenia z/OS .

Przegląd

Nazwa formatu: FMWIH.

Zestaw znaków i kodowanie: pola w strukturze MQWIH znajdują się w zestawie znaków i kodowaniu określonym przez pola *MDCSI* i *MDENC* w strukturze nagłówka poprzedzającej strukturę MQWIH lub przez te pola w strukturze MQMD, jeśli MQWIH znajduje się na początku danych komunikatu aplikacji.

Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach kolejek.

Użycie: Jeśli komunikat ma być przetwarzany przez menedżer obciążenia z/OS , musi zaczynać się od struktury MQWIH.

- "Pola" na stronie 1281
- "Wartości początkowe" na stronie 1283
- "Deklaracja RPG" na stronie 1283

Pola

Struktura MQWIH zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

WICSI (10-cyfrowa liczba całkowita ze znakiem)

Identyfikator zestawu znaków danych następujących po MQWIH.

Określa identyfikator zestawu znaków danych, które są zgodne ze strukturą MQWIH. Nie ma on zastosowania do danych znakowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Można użyć następującej wartości specjalnej:

CINHT
Dziedzicz identyfikator zestawu znaków tej struktury.

Dane znakowe w danych *następujących po* tej strukturze znajdują się w tym samym zestawie znaków co ta struktura.

Menedżer kolejek zmienia tę wartość w strukturze wysyłanej w komunikacie na rzeczywisty identyfikator zestawu znaków struktury. Jeśli nie wystąpi żaden błąd, wartość CSINHT nie jest zwracana przez wywołanie MQGET.

Parametr CSINHT nie może być używany, jeśli wartością pola *MDPAT* w MQMD jest ATBRKR.

Wartością początkową tego pola jest CSUNDF.

WIENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe danych następujących po MQWIH.

Określa kodowanie liczbowe danych, które są zgodne ze strukturą MQWIH. Nie ma ono zastosowania do danych liczbowych w samej strukturze MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych.

Wartością początkową tego pola jest 0.

WIFLG (10-cyfrowa liczba całkowita ze znakiem)

Flagi

Wartość musi być następująca:

WINON
Brak flag.

Wartością początkową tego pola jest WINONE.

WIFMT (8-bajtowy łańcuch znaków)

Nazwa formatu danych następujących po MQWIH.

Określa nazwę formatu danych, które są zgodne ze strukturą MQWIH.

W wywołaniu MQPUT lub MQPUT1 aplikacja musi ustawić w tym polu wartość odpowiednią dla danych. Reguły kodowania tego pola są takie same jak reguły kodowania pola *MDFMT* w strukturze MQMD.

Długość tego pola jest określona przez LNFMT. Wartością początkową tego pola jest FMNONE.

WILEN (10-cyfrowa liczba całkowita ze znakiem)

Długość struktury MQWIH.

Wartość musi być następująca:

WILEN1
Długość struktury nagłówka informacji o pracy w wersji version-1 .

Następująca stała określa długość bieżącej wersji:

WILENC
Długość bieżącej wersji struktury nagłówka informacji o pracy.

Wartością początkową tego pola jest WILEN1.

WIRSV (32-bajtowy łańcuch znaków)

Zarezerwowane.

Jest to pole zastrzeżone; musi być puste.

WISID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

WISIDV
Identyfikator struktury nagłówka informacji o pracy.

Wartością początkową tego pola jest WISIDV.

WISNM (32-bajtowy łańcuch znaków)

Nazwa usługi.

Jest to nazwa usługi, która ma przetworzyć komunikat.

Długość tego pola jest określona przez LNSVNM. Wartość początkowa tego pola to 32 znaki odstępu.

WISST (8-bajtowy łańcuch znaków)

Nazwa kroku usługi.

Jest to nazwa kroku *WISNM* , do którego odnosi się komunikat.

Długość tego pola jest określona przez LNSVST. Wartość początkowa tego pola to 8 znaków odstępu.

WITOK (16-bajtowy łańcuch bitowy)

Znacznik komunikatu.

Jest to znacznik komunikatu, który jednoznacznie identyfikuje komunikat.

W przypadku wywołań MQPUT i MQPUT1 to pole jest ignorowane. Długość tego pola jest określona przez LNMTOK. Wartością początkową tego pola jest MTKNON.

WIVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

WIVER1
Version-1 struktura nagłówka informacji o pracy.

Następująca stała określa numer wersji bieżącej:

WIVERC
Bieżąca wersja struktury nagłówka informacji o pracy.

Wartością początkową tego pola jest WIVER1.

Wartości początkowe

Tabela 739. Pola w MQWIH		
Nazwa pola	**Nazwa stałej**	**Wartość stałej**
WISID	WISIDV	'WIH¬'
WIVER	WIVER1	1
WILEN	WILEN1	120
WIENC	Brak	0
WICSI	CSUNDF	0
WIFMT	BRAK FMNONE	Puste
WIFLG	WINON	0
WISNM	Brak	Puste
WISST	Brak	Puste
WITOK	MMTKNON	Wartości null
WIRSV	Brak	Puste
Uwagi: 1. Symbol ¬ reprezentuje pojedynczy znak odstępu.		

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQWIH Structure
D*
D* Structure identifier
D  WISID                  1      4    INZ('WIH ')
D* Structure version number
D  WIVER                  5      8I 0 INZ(1)
D* Length of MQWIH structure
D  WILEN                  9     12I 0 INZ(120)
D* Numeric encoding of data that followsMQWIH
D  WIENC                 13     16I 0 INZ(0)
D* Character-set identifier of data thatfollows MQWIH
D  WICSI                 17     20I 0 INZ(0)
D* Format name of data that followsMQWIH
D  WIFMT                 21     28    INZ('        ')
D* Flags
D  WIFLG                 29     32I 0 INZ(0)
D* Service name
D  WISNM                 33     64    INZ
D* Service step name
D  WISST                 65     72    INZ
D* Message token
D  WITOK                 73     88    INZ(X'00000000000000-
D                                     000000000000000000')
D* Reserved
D  WIRSV                 89    120    INZ
```

IBM i MQXQH (nagłówek kolejki transmisji) w systemie IBM i

Struktura MQXQH opisuje informacje, które są poprzedzone danymi komunikatów aplikacji, gdy znajdują się one w kolejkach transmisji.

Przegląd

Cel: Kolejka transmisji jest specjalnym typem kolejki lokalnej, która tymczasowo przechowuje komunikaty przeznaczone dla kolejek zdalnych (czyli przeznaczone dla kolejek, które nie należą do menedżera kolejek lokalnych). Kolejka transmisji jest oznaczona atrybutem kolejki **Usage** o wartości USTRAN.

Nazwa formatu: FMXQH.

Zestaw znaków i kodowanie: Dane w MQXQH muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT dla języka programowania C.

Zestaw znaków i kodowanie MQXQH muszą być ustawione w polach *MDCSI* i *MDENC* w następujących polach:

- Oddzielna struktura MQMD (jeśli struktura MQXQH znajduje się na początku danych komunikatu) lub
- Struktura nagłówka poprzedzająca strukturę MQXQH (wszystkie inne przypadki).

Użycie: Komunikat znajdujący się w kolejce transmisji ma *dwa* deskryptory komunikatów:

- Jeden deskryptor komunikatu jest przechowywany oddzielnie od danych komunikatu. Jest on nazywany *oddzielnym deskryptorem komunikatu*i jest generowany przez menedżer kolejek, gdy komunikat jest umieszczany w kolejce transmisji. Niektóre pola w oddzielnym deskryptorze komunikatu są kopiowane z deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 .

  Osobny deskryptor komunikatu jest zwracany do aplikacji w parametrze **MSGDSC** wywołania MQGET, gdy komunikat jest usuwany z kolejki transmisji.
- Drugi deskryptor komunikatu jest przechowywany w strukturze MQXQH jako część danych komunikatu. Jest on nazywany *osadzonym deskryptorem komunikatu*i jest kopią deskryptora komunikatu udostępnionego przez aplikację w wywołaniu MQPUT lub MQPUT1 (z niewielkimi wariantami).

  Osadzonym deskryptorem komunikatu jest zawsze deskryptor MQMD version-1 . Jeśli komunikat umieszczony przez aplikację ma wartości inne niż domyślne dla jednego lub większej liczby pól version-2 w deskryptorze MQMD, struktura MQMDE jest zgodna z MQXQH i po niej następują dane komunikatu aplikacji (jeśli istnieją). Produkt MQMDE może mieć jedną z następujących wartości:

  – Wygenerowane przez menedżer kolejek (jeśli do umieszczenia komunikatu aplikacja używa deskryptora MQMD w wersji version-2) lub
  – Istnieje już na początku danych komunikatu aplikacji (jeśli do umieszczenia komunikatu aplikacja używa programu MQMD w wersji version-1).

  Osadzony deskryptor komunikatu jest to deskryptor, który jest zwracany do aplikacji w parametrze **MSGDSC** wywołania MQGET, gdy komunikat jest usuwany z końcowej kolejki docelowej.

Pola w oddzielnym deskryptorze komunikatu

Pola w oddzielnym deskryptorze komunikatu są ustawiane przez menedżer kolejek w sposób przedstawiony na poniższej liście. Jeśli menedżer kolejek nie obsługuje deskryptora MQMD version-2 , używany jest deskryptor MQMD version-1 bez utraty funkcji.

Tabela 740. Pola w oddzielnym deskryptorze komunikatu i używane wartości

Pole w oddzielnej strukturze MQMD	Użyta wartość
MDSID	MDSIDV
MDVER	MDVER2
MDREP	Kopiowane z osadzonego deskryptora komunikatu, ale z bitami identyfikowanymi przez ROAUXM ustawionymi na zero. Zapobiega to generowaniu komunikatu raportu COA lub COD, gdy komunikat jest umieszczany w kolejce transmisji lub z niej usuwany.
MDMT	Skopiowane z osadzonego deskryptora komunikatu.
MDEXP	Skopiowane z osadzonego deskryptora komunikatu.
MDFB	Skopiowane z osadzonego deskryptora komunikatu.
MDENC	ENNAT,
MDCSI	Atrybut **CodedCharSetId** menedżera kolejek.
MDFMT	FMXQH
MDPRI	Skopiowane z osadzonego deskryptora komunikatu.
MDPER	Skopiowane z osadzonego deskryptora komunikatu.
MDMID	Nowa wartość jest generowana przez menedżer kolejek. Ten identyfikator komunikatu różni się od identyfikatora komunikatu *MDMID* , który mógł zostać wygenerowany przez menedżer kolejek dla osadzonego deskryptora komunikatu (patrz opis powyżej).
MDCID	Wartość *MDMID* z osadzonego deskryptora komunikatu.
MDBOC	0
MDRQ	Skopiowane z osadzonego deskryptora komunikatu.
MDRM	Skopiowane z osadzonego deskryptora komunikatu.
MDUID	Skopiowane z osadzonego deskryptora komunikatu.
MDACC	Skopiowane z osadzonego deskryptora komunikatu.
MDAID	Skopiowane z osadzonego deskryptora komunikatu.
MDPAT	ATQM,
MDPAN	Pierwsze 28 bajtów nazwy menedżera kolejek.
MDPD	Data umieszczenia komunikatu w kolejce transmisji.
MDPT	Czas umieszczenia komunikatu w kolejce transmisji.
MDAOD	Puste
MDGID	GINON
MDSEQ	1
MDOFF	0
MDMFL	BRAK MFNONE

Tabela 740. Pola w oddzielnym deskryptorze komunikatu i używane wartości (kontynuacja)

Pole w oddzielnej strukturze MQMD	Użyta wartość
MDOLN	OLUNDF

Pola w osadzonym deskryptorze komunikatu

Pola w osadzonym deskryptorze komunikatu mają takie same wartości jak pola w parametrze **MSGDSC** wywołania MQPUT lub MQPUT1 , z wyjątkiem następujących:

- Pole *MDVER* zawsze ma wartość MDVER1.
- Jeśli pole *MDPRI* ma wartość PRQDEF, jest ono zastępowane wartością atrybutu **DefPriority** kolejki.
- Jeśli pole *MDPER* ma wartość PEQDEF, jest ono zastępowane wartością atrybutu **DefPersistence** kolejki.
- Jeśli pole *MDMID* ma wartość MINONE lub podano opcję PMNMID albo komunikat jest komunikatem listy dystrybucyjnej, wartość *MDMID* jest zastępowana nowym identyfikatorem komunikatu wygenerowanym przez menedżer kolejek.

  Gdy komunikat listy dystrybucyjnej jest dzielony na mniejsze komunikaty listy dystrybucyjnej umieszczone w różnych kolejkach transmisji, pole *MDMID* w każdym z nowych osadzonych deskryptorów komunikatów jest takie samo jak w oryginalnym komunikacie listy dystrybucyjnej.
- Jeśli podano opcję PMNCID, wartość *MDCID* jest zastępowana nowym identyfikatorem korelacji wygenerowanym przez menedżer kolejek.
- Pola kontekstu są ustawiane zgodnie z opcjami PM* określonymi w parametrze **PMO** . Pola kontekstu są następujące:
  - *MDACC*
  - *MDAID*
  - *MDAOD*
  - *MDPAN*
  - *MDPAT*
  - *MDPD*
  - *MDPT*
  - *MDUID*
- Pola version-2 (jeśli były obecne) są usuwane z deskryptora MQMD i przenoszone do struktury MQMDE, jeśli co najmniej jedno z pól version-2 ma wartość inną niż domyślna.

Umieszczanie komunikatów w kolejkach zdalnych

Gdy aplikacja umieszcza komunikat w kolejce zdalnej (przez bezpośrednie określenie nazwy kolejki zdalnej lub użycie lokalnej definicji kolejki zdalnej), menedżer kolejek lokalnych:

- Tworzy strukturę MQXQH zawierającą osadzony deskryptor komunikatu
- Dołącza MQMDE, jeśli jest potrzebny i nie jest jeszcze obecny
- Dołącza dane komunikatu aplikacji
- Umieszcza komunikat w odpowiedniej kolejce transmisji

Umieszczanie komunikatów bezpośrednio w kolejkach transmisji

Możliwe jest również, że aplikacja umieściła komunikat bezpośrednio w kolejce transmisji. W takim przypadku aplikacja musi poprzedzać dane komunikatu aplikacji strukturą MQXQH i inicjować pola odpowiednimi wartościami. Ponadto pole *MDFMT* w parametrze **MSGDSC** wywołania MQPUT lub MQPUT1 musi mieć wartość FMXQH.

Dane znakowe w strukturze MQXQH utworzonej przez aplikację muszą znajdować się w zestawie znaków lokalnego menedżera kolejek (zdefiniowanym przez atrybut menedżera kolejek **CodedCharSetId**), a dane całkowite muszą być zakodowane na komputerze rodzimym. Ponadto dane znakowe w strukturze MQXQH muszą być dopełnione odstępami do zdefiniowanej długości pola. Dane nie mogą zostać przedwcześnie zakończone przy użyciu znaku o kodzie zero, ponieważ menedżer kolejek nie przekształca znaków o kodzie zero i kolejnych znaków w znaki o kodzie zero w strukturze MQXQH.

Należy jednak zauważyć, że menedżer kolejek nie sprawdza, czy istnieje struktura MQXQH lub czy dla pól określono poprawne wartości.

Pobieranie komunikatów z kolejek transmisji

Aplikacje, które pobierają komunikaty z kolejki transmisji, muszą przetwarzać informacje w strukturze MQXQH w odpowiedni sposób. Obecność struktury MQXQH na początku danych komunikatu aplikacji jest wskazywana przez wartość FMXQH zwracaną w polu *MDFMT* parametru **MSGDSC** wywołania MQGET. Wartości zwracane w polach *MDCSI* i *MDENC* parametru **MSGDSC** wskazują zestaw znaków i kodowanie danych znakowych i całkowitych w strukturze MQXQH. Zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane przez pola *MDCSI* i *MDENC* w osadzonym deskryptorze komunikatu.

Pola

Struktura MQXQH zawiera następujące pola; pola są opisane w **porządku alfabetycznym**:

XQMD (MQMD1)

Oryginalny deskryptor komunikatu.

Jest to osadzony deskryptor komunikatu i jest to bliska kopia deskryptora komunikatu MQMD, który został określony jako parametr **MSGDSC** w wywołaniu MQPUT lub MQPUT1 podczas pierwotnego umieszczania komunikatu w kolejce zdalnej.

Uwaga: Jest to komenda MQMD w wersji version-1 .

Wartości początkowe pól w tej strukturze są takie same jak w strukturze MQMD.

XQRQ (48-bajtowy łańcuch znaków)

Nazwa kolejki docelowej.

Jest to nazwa kolejki komunikatów, która jest pozornym miejscem docelowym komunikatu (może się okazać, że nie jest to rzeczywiste miejsce docelowe, jeśli na przykład ta kolejka jest zdefiniowana w *XQRQM* jako lokalna definicja innej kolejki zdalnej).

Jeśli komunikat jest komunikatem listy dystrybucyjnej (pole *MDFMT* w deskryptorze komunikatu osadzonego to FMDH), pole *XQRQ* jest puste.

Długość tego pola jest określona przez LNQN. Wartością początkową tego pola jest 48 znaków odstępu.

XQRQM (48-bajtowy łańcuch znaków)

Nazwa docelowego menedżera kolejek.

Jest to nazwa menedżera kolejek lub grupy współużytkowania kolejek, która jest właścicielem kolejki będącej pozornym miejscem docelowym komunikatu.

Jeśli komunikat jest komunikatem listy dystrybucyjnej, pole *XQRQM* jest puste.

Długość tego pola jest określona przez LNQMN. Wartością początkową tego pola jest 48 znaków odstępu.

XQSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

XQSIDV
Identyfikator struktury nagłówka kolejki transmisji.

Wartością początkową tego pola jest XQSIDV.

XQVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

XQVER1
Numer wersji struktury nagłówka kolejki transmisji.

Następująca stała określa numer wersji bieżącej:

XQVERC
Bieżąca wersja struktury nagłówka kolejki transmisji.

Wartością początkową tego pola jest XQVER1.

Wartości początkowe

Tabela 741. Pola w MQXQH

Nazwa pola	Nazwa stałej	Wartość stałej
XQSID	XQSIDV	'XQH¬'
XQVER	XQVER1	1
XQRQ	Brak	Puste
XQRQM	Brak	Puste
XQMD	Takie same nazwy i wartości jak w przypadku deskryptora MQMD (patrz sekcja Tabela 709 na stronie 1185).	-

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.

Deklaracja RPG

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D*
D* MQXQH Structure
D*
D* Structure identifier
D  XQSID                  1      4    INZ('XQH ')
D* Structure version number
D  XQVER                  5      8I 0 INZ(1)
D* Name of destination queue
D  XQRQ                   9     56    INZ
D* Name of destination queue manager
D  XQRQM                 57    104    INZ
D* Original message descriptor
D  XQ1SID               105    108    INZ('MD  ')
D  XQ1VER               109    112I 0 INZ(1)
D  XQ1REP               113    116I 0 INZ(0)
D  XQ1MT                117    120I 0 INZ(8)
D  XQ1EXP               121    124I 0 INZ(-1)
D  XQ1FB                125    128I 0 INZ(0)
D  XQ1ENC               129    132I 0 INZ(273)
D  XQ1CSI               133    136I 0 INZ(0)
D  XQ1FMT               137    144    INZ('        ')
D  XQ1PRI               145    148I 0 INZ(-1)
D  XQ1PER               149    152I 0 INZ(2)
D  XQ1MID               153    176    INZ(X'00000000000000-
D                                     000000000000000000000000-
```

```
D                                       000000000000')
D  XQ1CID               177    200     INZ(X'00000000000000-
D                                       00000000000000000000-
D                                       000000000000')
D  XQ1BOC               201    204I 0  INZ(0)
D  XQ1RQ                205    252     INZ
D  XQ1RM                253    300     INZ
D  XQ1UID               301    312     INZ
D  XQ1ACC               313    344     INZ(X'00000000000000-
D                                       00000000000000000000-
D                                       00000000000000000000-
D                                       000000')
D  XQ1AID               345    376     INZ
D  XQ1PAT               377    380I 0  INZ(0)
D  XQ1PAN               381    408     INZ
D  XQ1PD                409    416     INZ
D  XQ1PT                417    424     INZ
D  XQ1AOD               425    428     INZ
```

IBM i Wywołania funkcji w systemie IBM i

Ten temat zawiera informacje o wywołaniach funkcji dostępnych w programie IBM i .

Konwencje stosowane w opisach wywołań w systemie IBM i

Dla każdego wywołania ta kolekcja tematów zawiera opis parametrów i sposobu użycia wywołania. Po tym następują typowe wywołania wywołania i typowe deklaracje jego parametrów w języku programowania RPG.

Ważne: Podczas kodowania wywołań funkcji API języka IBM MQ należy upewnić się, że zostały podane wszystkie istotne parametry (zgodnie z opisem w poniższych sekcjach). Brak takiego działania może spowodować nieprzewidywalne rezultaty.

Opis każdego wywołania zawiera następujące sekcje:

Nazwa połączenia
Nazwa połączenia, po której następuje krótki opis przeznaczenia połączenia.

Parametry
Dla każdego parametru po nazwie występuje jego typ danych w nawiasach () i jego kierunek, na przykład:

CMPCOD (9-cyfrowa liczba dziesiętna)-dane wyjściowe

Więcej informacji na temat typów danych struktury zawiera sekcja “Podstawowe typy danych” na stronie 1025.

Kierunek parametru może być następujący:

Wejście
Użytkownik (programista) musi podać ten parametr.

Wyjście
Wywołanie zwraca ten parametr.

Wejście/wyjście
Należy podać ten parametr, ale jest on modyfikowany przez wywołanie.

Dostępny jest również krótki opis przeznaczenia parametru wraz z listą wartości, które mogą być używane przez ten parametr.

Dwa ostatnie parametry w każdym wywołaniu są kodem zakończenia i kodem przyczyny. Kod zakończenia wskazuje, czy wywołanie zakończyło się pomyślnie, częściowo, czy wcale. Więcej informacji o częściowym powodzeniu lub niepowodzeniu wywołania znajduje się w kodzie przyczyny.

Użycie notatek
Dodatkowe informacje o wywołaniu, opisujące sposób jego użycia oraz wszelkie ograniczenia dotyczące jego użycia.

Wywołanie RPG
Typowe wywołanie wywołania i deklaracja jego parametrów w języku RPG.

Inne konwencje notacyjne to:

Stałe
Nazwy stałych są wyświetlane wielkimi literami, na przykład OOOUT.

Tablice
W niektórych wywołaniach parametry są tablicami łańcuchów znaków o nieustalonej wielkości. W opisach tych parametrów mała litera *n* oznacza stałą numeryczną. Podczas kodowania deklaracji tego parametru należy zastąpić wartość *n* wymaganą wartością liczbową.

IBM i MQBACK (wycofanie zmian) w systemie IBM i

Wywołanie MQBACK wskazuje menedżerowi kolejek, że wszystkie operacje pobierania i umieszczania komunikatów, które wystąpiły od ostatniego punktu synchronizacji, mają zostać wycofane. Komunikaty umieszczone jako część jednostki pracy są usuwane; komunikaty pobrane jako część jednostki pracy są przywracane do kolejki.

- To wywołanie jest obsługiwane w następujących środowiskach:
  - AIX AIX
  - IBM i IBM i
  - Windows Windows
- “Składnia” na stronie 1290
- “Użycie notatek” na stronie 1290
- “Parametry” na stronie 1292
- “Deklaracja RPG” na stronie 1292

Składnia

MQBACK *(Hconn, CompCode, Reason)*

Użycie notatek

Podczas używania komendy MQBACK należy wziąć pod uwagę następujące uwagi dotyczące użycia.

1. Tego wywołania można użyć tylko wtedy, gdy sam menedżer kolejek koordynuje jednostkę pracy. Jest to lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiedniego wywołania wycofania zamiast wywołania MQBACK. Środowisko może również obsługiwać niejawne wycofanie spowodowane nieprawidłowym zakończeniem działania aplikacji.
   - W systemie IBM itego wywołania można użyć dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie została wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji “MQDISC (Rozłączenie menedżera kolejek) w systemie IBM i” na stronie 1330 .
4. Gdy aplikacja umieszcza lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla

ostatnich pomyślnych wywołań MQPUT i MQGET. Te informacje są powiązane z uchwytem kolejki i obejmują takie elementy, jak:

- Wartości pól *MDGID*, *MDSEQ*, *MDOFF*i *MDMFL* w strukturze MQMD.
- Określa, czy komunikat jest częścią jednostki pracy.
- Dla wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Menedżer kolejek przechowuje *trzy* zestawy informacji o grupach i segmentach, po jednym zestawie dla każdego z następujących zestawów:

- Ostatnie pomyślne wywołanie MQPUT (może być częścią jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które przeglądnięło komunikat w kolejce (nie może być częścią jednostki pracy).

Jeśli aplikacja umieszcza lub pobiera komunikaty jako część jednostki pracy, a następnie decyduje się wycofać jednostkę pracy, informacje o grupie i segmencie są przywracane do wartości, która była wcześniej używana:

- Informacje powiązane z wywołaniem MQPUT zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQPUT dla tego uchwytu kolejki w bieżącej jednostce pracy.
- Informacje powiązane z wywołaniem MQGET zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zasięgiem jednostki pracy, nie mają odtworzonych informacji o grupie i segmencie, jeśli jednostka pracy została wycofana.

Przywrócenie poprzedniej wartości informacji o grupie i segmencie po wycofaniu jednostki pracy umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów na kilka jednostek pracy oraz zrestartowanie w odpowiednim punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się. Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną pamięć kolejki. Jednak aplikacja musi zachować wystarczającą ilość informacji, aby w przypadku awarii systemu możliwe było zrestartowanie umieszczania lub pobierania komunikatów w odpowiednim miejscu. Szczegółowe informacje na temat restartowania w odpowiednim punkcie po awarii systemu zawiera opis opcji PMLOGO w sekcji “MQPMO (opcje umieszczania komunikatów) w systemie IBM i” na stronie 1208oraz opis opcji GMLOGO w sekcji “MQGMO (opcje pobierania komunikatów) w systemie IBM i” na stronie 1106.

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

1. Jednostka pracy ma taki sam zasięg jak uchwyt połączenia. Oznacza to, że wszystkie wywołania IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wykonane przy użyciu innego uchwytu połączenia (na przykład wywołania wykonane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączeń zawiera opis parametru **HCONN** w sekcji “MQCONN (połączenie z menedżerem kolejek) w systemie IBM i” na stronie 1316 .
2. To wywołanie ma wpływ tylko na komunikaty, które zostały umieszczone lub pobrane jako część bieżącej jednostki pracy.
3. Aplikacja długotrwała, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w jednostce pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może spowodować zapełnienie kolejek komunikatami, które nie są dostępne dla innych aplikacji. W celu zabezpieczenia się przed taką możliwością administrator powinien ustawić atrybut menedżera kolejek **MaxUncommittedMsgs** na wartość, która jest na tyle niska, aby zapobiec zapełnianiu kolejek przez aplikacje, ale na tyle wysoka, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQBACK ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *COMCOD*.

Jeśli *COMCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *COMCOD* to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2123
(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQBACK(HCONN : COMCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
```

```
     DMQBACK           PR                  EXTPROC('MQBACK')
     D* Connection handle
     D HCONN                        10I 0 VALUE
     D* Completion code
     D COMCOD                       10I 0
     D* Reason code qualifying COMCOD
     D REASON                       10I 0
```

IBM i MQBEGIN (rozpoczęcie jednostki pracy) w systemie IBM i

Wywołanie komendy MQBEGIN rozpoczyna jednostkę pracy, która jest koordynowana przez menedżer kolejek i która może obejmować zewnętrzne menedżery zasobów.

- To wywołanie jest obsługiwane w następujących środowiskach:
  - AIX AIX
  - IBM i IBM i
  - Windows Windows
- “Składnia” na stronie 1293
- “Użycie notatek” na stronie 1293
- “Parametry” na stronie 1294
- “Deklaracja RPG” na stronie 1295

Składnia

MQBEGIN *(HCONN, BEGOP, CMPCOD, REASON)*

Użycie notatek

1. Wywołania MQBEGIN można użyć do uruchomienia jednostki pracy koordynowanej przez menedżer kolejek, która może obejmować zmiany zasobów należących do innych menedżerów zasobów. Menedżer kolejek obsługuje trzy typy jednostek pracy:

   Menedżer kolejek-skoordynowana lokalna jednostka pracy
   Jest to jednostka pracy, w której menedżer kolejek jest jedynym uczestniczącym menedżerem zasobów, dlatego menedżer kolejek pełni rolę koordynatora jednostki pracy.

   - Aby uruchomić ten typ jednostki pracy, należy określić opcję PMSYP lub GMSYP w pierwszym wywołaniu MQPUT, MQPUT1lub MQGET w jednostce pracy.

     Aplikacja nie musi wywoływać wywołania MQBEGIN w celu uruchomienia jednostki pracy, ale jeśli zostanie użyta opcja MQBEGIN, wywołanie zostanie zakończone z kodem CCWARN i kodem przyczyny RC2121.
   - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołania MQCMIT lub MQBACK.

   Menedżer kolejek-skoordynowana globalna jednostka pracy
   Jest to jednostka pracy, w której menedżer kolejek pełni rolę koordynatora jednostki pracy, zarówno dla IBM MQ zasobów , *jak i* dla zasobów należących do innych menedżerów zasobów. Te menedżery zasobów współpracują z menedżerem kolejek w celu zapewnienia, że wszystkie zmiany zasobów w jednostce pracy zostaną zatwierdzone lub wycofane razem.

   - Aby uruchomić ten typ jednostki pracy, należy użyć wywołania MQBEGIN.
   - Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań MQCMIT i MQBACK.

Globalna jednostka pracy koordynowana zewnętrznie
Jest to jednostka pracy, w której menedżer kolejek jest uczestnikiem, ale menedżer kolejek nie działa jako koordynator jednostki pracy. Zamiast tego istnieje zewnętrzny koordynator jednostki pracy, z którym współpracuje menedżer kolejek.

- Aby uruchomić ten typ jednostki pracy, należy użyć odpowiedniego wywołania udostępnionego przez zewnętrznego koordynatora jednostki pracy.

  Jeśli wywołanie komendy MQBEGIN jest używane do próby uruchomienia jednostki pracy, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2012.
- Aby zatwierdzić lub wycofać ten typ jednostki pracy, należy użyć wywołań zatwierdzenia i potwierdzenia dostarczonych przez zewnętrznego koordynatora jednostki pracy.

  Jeśli wywołanie MQCMIT lub MQBACK jest używane do zatwierdzania lub wycofywania jednostki pracy, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2012.

2. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji "MQDISC (Rozłączenie menedżera kolejek) w systemie IBM i" na stronie 1330 .
3. Aplikacja może jednocześnie uczestniczyć tylko w jednej jednostce pracy. Wywołanie komendy MQBEGIN kończy się niepowodzeniem z kodem przyczyny RC2128 , jeśli dla aplikacji istnieje już jednostka pracy, niezależnie od typu tej jednostki pracy.
4. Wywołanie MQBEGIN nie jest poprawne w środowisku klienta IBM MQ . Próba użycia wywołania nie powiodła się z kodem przyczyny RC2012.
5. Jeśli menedżer kolejek działa jako koordynator jednostki pracy dla globalnych jednostek pracy, menedżery zasobów, które mogą uczestniczyć w tej jednostce pracy, są definiowane w pliku konfiguracyjnym menedżera kolejek.
6. W systemie IBM iobsługiwane są trzy typy jednostek pracy:

   - Opcja **Lokalne jednostki pracy koordynowane przez menedżer kolejek** może być używana tylko wtedy, gdy definicja kontroli transakcji nie istnieje na poziomie zadania, tzn. komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie może być wydana dla zadania.
   - **Globalne jednostki pracy koordynowane przez menedżer kolejek** nie są obsługiwane.
   - **Globalne jednostki pracy koordynowane zewnętrznie** mogą być używane tylko wtedy, gdy definicja kontroli transakcji istnieje na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** musi zostać wydana dla zadania. Jeśli zostało to zrobione, operacje IBM i COMMIT i ROLLBACK mają zastosowanie do zasobów IBM MQ , a także do zasobów należących do innych uczestniczących menedżerów zasobów.

Parametry

Wywołanie MQBEGIN ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

BEGOP (MQBO)-wejście/wyjście

Opcje sterujące działaniem komendy MQBEGIN.

Szczegółowe informacje można znaleźć w sekcji "MQBO (opcje początku) w systemie IBM i" na stronie 1047.

Jeśli nie są wymagane żadne opcje, programy napisane w języku C lub S/390 mogą określać adres parametru o wartości NULL zamiast określać adres struktury MQBO.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2121
(2121, X'849 ') Brak zarejestrowanych uczestniczących menedżerów zasobów.

RC2122
(2122, X'84A') Uczestniczy menedżer zasobów jest niedostępny.

Jeśli *CMPCOD* to CCFAIL:

RC2134
(2134, X'856 ') Niepoprawna struktura opcji początku.

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2012
(2012, X'7DC') Wywołanie niepoprawne w środowisku.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2128
(2128, X'850 ') Jednostka pracy została już uruchomiona.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
```

```
C                     CALLP     MQBEGIN(HCONN : BEGOP : CMPCOD :
C                                       REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQBEGIN          PR                  EXTPROC('MQBEGIN')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Options that control the action of MQBEGIN
D BEGOP                         12A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQBUFMH (Convert buffer into message handle-Przekształć bufor w uchwyt komunikatu) w systemie IBM i

Wywołanie funkcji MQBUFMH przekształca bufor w uchwyt komunikatu i jest odwrotnością wywołania MQMHBUF.

To wywołanie pobiera deskryptor komunikatu i właściwości MQRFH2 w buforze i udostępnia je za pośrednictwem uchwytu komunikatu. Właściwości MQRFH2 w danych komunikatu są opcjonalnie usuwane. Pola *Encoding*, *CodedCharSetId*i *Format* deskryptora komunikatu są aktualizowane, jeśli jest to konieczne, w celu poprawnego opisania zawartości buforu po usunięciu właściwości.

- "Składnia" na stronie 1296
- "Użycie notatek" na stronie 1296
- "Parametry" na stronie 1296
- "Deklaracja RPG" na stronie 1298

Składnia

MQBUFMH *(Hconn, Hmsg, BufMsgHOpts, MsgDesc, Buffer, BufferLength, DataLength, CompCode, Reason)*

Użycie notatek

Wywołania MQBUFMH nie mogą być przechwytywane przez wyjścia API-bufor jest przekształcany w uchwyt komunikatu w obszarze aplikacji; wywołanie nie dociera do menedżera kolejek.

Parametry

Wywołanie MQBUFMH ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony za pomocą HCUNAS, należy nawiązać poprawne połączenie w wątku, przekształcając bufor w uchwyt komunikatu. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt jest uchwytem komunikatu, dla którego wymagany jest bufor. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

BMHOPT (MQBMHO)-wejście

Struktura MQBMHO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia uchwytów komunikatów z buforów.

Szczegółowe informacje można znaleźć w sekcji “MQBMHO (opcje od buforu do uchwytu komunikatu) w systemie IBM i” na stronie 1045.

MSGDSC (MQMD)-wejście/wyjście

Struktura *MSGDSC* zawiera właściwości deskryptora komunikatu i opisuje treść obszaru buforu.

W danych wyjściowych wywołania właściwości są opcjonalnie usuwane z obszaru buforu i w tym przypadku deskryptor komunikatu jest aktualizowany w celu poprawnego opisania obszaru buforu.

Dane w tej strukturze muszą być w zestawie znaków i kodowaniu aplikacji.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

BUFLEN to długość obszaru buforu w bajtach.

Wartość *BUFLEN* równa zero bajtów jest poprawna i wskazuje, że obszar buforu nie zawiera danych.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście/wyjście

BUFFER definiuje obszar zawierający bufor komunikatów. W przypadku większości danych należy wyrównać bufor do granicy 4-bajtowej.

Jeśli plik *BUFFER* zawiera dane znakowe lub liczbowe, należy ustawić pola *CodedCharSetId* i *Encoding* w parametrze **MSGDSC** na wartości odpowiednie dla danych. W razie potrzeby umożliwia to konwersję danych.

Jeśli w buforze komunikatów zostaną znalezione właściwości, zostaną one opcjonalnie usunięte. Później staną się dostępne w uchwycie komunikatu po powrocie z wywołania.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void, co oznacza, że jako parametr można podać adres dowolnego typu danych.

Jeśli parametr **BUFLEN** ma wartość zero, oznacza to, że nie występuje odwołanie do parametru *BUFFER* . W takim przypadku adres parametru przekazywany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość NULL.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

DATLEN jest długością (w bajtach) buforu, z którego mogły zostać usunięte właściwości.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2489
(2489, X'09B9') Niepoprawna struktura opcji uchwytu buforu do komunikatu.

RC2004
(2004, X'07D4') Niepoprawny parametr buforu.

RC2005
(2005, X'07D5') Niepoprawny parametr długości buforu.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2460
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

RC2026
(2026, X'07EA') Niepoprawny deskryptor komunikatu.

RC2499
(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

RC2334
(2334, X'091E') Niepoprawna struktura MQRFH2 .

RC2421
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQBUFMH(HCONN : HMSG : BMHOPT :
                                        MSGDSC : BUFLEN : BUFFER :
                                        DATLEN : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQBUFMH          PR                  EXTPROC('MQBUFMH')
D* Connection handle
D HCONN                         10I 0
D* Message handle
D HMSG                          10I 0
D* Options that control the action of MQBUFMH
D BMHOPT                        12A   VALUE
D* Message descriptor
D MSGDSC                       364A
D* Length in bytes of the Buffer area
D BUFLEN                        10I 0
D* Area to contain the message buffer
D BUFFER                          *   VALUE
D* Length of the output buffer
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQCB (Manage callback) w systemie IBM i

Wywołanie MQCB ponownie zarejestrowyje wywołanie zwrotne dla określonego uchwytu obiektu i steruje aktywacją i zmianami wywołania zwrotnego.

Wywołanie zwrotne jest fragmentem kodu (określanym jako nazwa funkcji, która może być dowiązana dynamicznie lub jako wskaźnik funkcji), który jest wywoływany przez funkcję IBM MQ w przypadku wystąpienia określonych zdarzeń.

Aby używać obiektów MQCB i MQCTL na kliencie V7 , należy nawiązać połączenie z serwerem V7 , a parametr **SHARECNV** kanału musi mieć wartość niezerową.

Więcej informacji na temat globalnych jednostek pracy zawiera sekcja Globalne jednostki pracy.

Można zdefiniować następujące typy wywołań zwrotnych:

Konsument komunikatu
Funkcja zwrotna konsumenta komunikatów jest wywoływana, gdy komunikat spełniający określone kryteria wyboru jest dostępny w uchwycie obiektu.

Dla każdego uchwytu obiektu można zarejestrować tylko jedną funkcję zwrotną. Jeśli jedna kolejka ma być odczytywana z wieloma kryteriami wyboru, kolejka musi być otwierana wiele razy, a funkcja konsumenta musi być zarejestrowana w każdym uchwycie.

procedura obsługi zdarzeń
Procedura obsługi zdarzeń jest wywoływana dla warunków, które mają wpływ na całe środowisko wywołania zwrotnego.

Funkcja jest wywoływana w przypadku wystąpienia warunku zdarzenia, na przykład zatrzymania lub wyciszenia menedżera kolejek lub połączenia.

Funkcja nie jest wywoływana dla warunków specyficznych dla pojedynczego konsumenta komunikatów, na przykład RC2016; jest wywoływana, jeśli funkcja zwrotna nie zakończy się normalnie.

- "Składnia" na stronie 1299
- "Uwagi dotyczące użycia obiektu MQCB" na stronie 1299
- "Parametry dla obiektu MQCB" na stronie 1301
- "Deklaracja RPG" na stronie 1307

Składnia

Obiekt MQCB *(HCONN, OPERATN, HOBJ, CBDSC, MSGDSC, GMO, CMPCOD, REASON)*

Uwagi dotyczące użycia obiektu MQCB

1. Obiekt MQCB jest używany do definiowania działania, które ma być wywoływane dla każdego komunikatu, zgodnego z określonymi kryteriami, dostępnego w kolejce. Podczas przetwarzania działania komunikat jest usuwany z kolejki i przekazywany do zdefiniowanego konsumenta komunikatów lub udostępniany jest znacznik komunikatu, który jest używany do pobierania komunikatu.
2. Obiekt MQCB może być używany do definiowania podprogramów wywołania zwrotnego przed rozpoczęciem korzystania z obiektu MQCTL lub może być używany z poziomu podprogramu wywołania zwrotnego.
3. Aby użyć obiektu MQCB spoza procedury wywołania zwrotnego, należy najpierw zawiesić przetwarzanie komunikatów przy użyciu obiektu MQCTL i wznowić przetwarzanie.

Sekwencja wywołań zwrotnych konsumenta komunikatów

Konsument można skonfigurować w taki sposób, aby wywoływał wywołanie zwrotne w punktach kluczowych podczas cyklu życia konsumenta. Na przykład:

- gdy konsument jest po raz pierwszy zarejestrowany,
- po uruchomieniu połączenia,
- po zatrzymaniu połączenia i
- gdy konsument jest wyrejestrowany jawnie lub niejawnie przez MQCLOSE.

Tabela 742. Definicje komend MQCTL

Czasownik	Znaczenie
MQCTL (URUCHOM)	Wywołanie MQCTL z użyciem operacji CTLSR
MQCTL (ZATRZYMAJ)	Wywołanie MQCTL przy użyciu operacji CTLSP
MQCTL (OCZEKIWANIE)	Wywołanie MQCTL przy użyciu operacji CTLSW

Umożliwia konsumentowi zachowanie stanu powiązanego z konsumentem. Gdy aplikacja żąda wywołania zwrotnego, reguły wywołania konsumenta są następujące:

ZAREJESTRUJ SIĘ
Jest zawsze pierwszym typem wywołania wywołania zwrotnego.

Jest zawsze wywoływana w tym samym wątku, co wywołanie CBREG (MQCB).

START
Jest zawsze wywoływana synchronicznie z komendą MQCTL (START).

- Wszystkie wywołania zwrotne komendy START są wykonywane przed zwróceniem komendy MQCTL (START).

W przypadku żądania CTLTHR znajduje się w tym samym wątku, co dostarczanie wiadomości.

Wywołanie z komendą start nie jest gwarantowane, jeśli na przykład poprzednie wywołanie zwrotne wywołuje komendę MQCTL (STOP) podczas wykonywania komendy MQCTL (START).

STOP
Kolejne komunikaty i zdarzenia nie będą dostarczane po tym wywołaniu, dopóki połączenie nie zostanie zrestartowane.

Komenda STOP jest gwarantowana, jeśli aplikacja została wcześniej wywołana dla komendy START, komunikatu lub zdarzenia.

DEREGISTER (Deregator)
Jest zawsze ostatnim typem wywołania wywołania zwrotnego.

Upewnij się, że aplikacja wykonuje inicjowanie i czyszczenie w oparciu o wątki w wywołaniach zwrotnych START i STOP. Za pomocą wywołań zwrotnych REGISTER i DEREGISTER można inicjować i wykonywać procedury czyszczące, które nie są oparte na wątkach.

Nie należy zakładać, że czas życia i dostępność wątku jest inna niż to, co zostało określone. Na przykład nie należy polegać na wątku, który pozostaje aktywny po ostatnim wywołaniu funkcji DEREGISTER. Podobnie, jeśli nie wybrano opcji CTLTHR, nie należy zakładać, że wątek istnieje przy każdym uruchomieniu połączenia.

Jeśli aplikacja ma określone wymagania dotyczące parametrów wątków, zawsze może utworzyć odpowiedni wątek, a następnie użyć komendy MQCTL (WAIT). Ten krok *przekazuje* wątek do programu IBM MQ w celu asynchronicznego dostarczania komunikatów.

Użycie połączenia konsumenta komunikatów

Zwykle, gdy aplikacja wysyła kolejne wywołanie MQI, gdy jedno z nich jest zaległe, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2219.

Istnieją jednak specjalne przypadki, w których aplikacja musi wykonać kolejne wywołanie MQI przed zakończeniem poprzedniego wywołania. Na przykład konsument może być wywoływany podczas wywołania MQCB za pomocą komendy CBRE.

W takiej sytuacji, gdy w wyniku wydania przez aplikację komendy MQCB lub MQCTL aplikacja jest wywoływana z powrotem, może ona wykonać kolejne wywołanie MQI. Ta instancja oznacza, że można wywołać na przykład wywołanie MQOPEN w funkcji konsumenta po wywołaniu CBCCALLT typu CBCTRC. Dozwolone jest każde wywołanie MQI z wyjątkiem wywołania MQDISC.

Parametry dla obiektu MQCB

Wywołanie MQCB ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Zarządzaj funkcją zwrotną-parametr HCONN.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

OPERATN (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Zarządzanie funkcją zwrotną-parametr OPERATN.

Operacja przetwarzana w wywołaniu zwrotnym zdefiniowanym dla określonego uchwytu obiektu. Należy określić jedną z następujących opcji; jeśli wymagana jest więcej niż jedna opcja, wartości mogą być dodawane (nie należy dodawać tej samej stałej więcej niż raz) lub łączone za pomocą operacji bitowej OR (jeśli język programowania obsługuje operacje bitowe).

Kombinacje, które nie są poprawne, są odnotowywane; wszystkie pozostałe kombinacje są poprawne.

CBREG
Zdefiniuj funkcję zwrotną dla określonego uchwytu obiektu. Ta operacja definiuje funkcję, która ma zostać wywołana, oraz kryteria wyboru, które mają zostać użyte.

Jeśli funkcja zwrotna jest już zdefiniowana dla uchwytu obiektu, definicja jest zastępowana. Jeśli podczas zastępowania wywołania zwrotnego zostanie wykryty błąd, funkcja zostanie wyrejestrowana.

Jeśli wywołanie zwrotne jest zarejestrowane w tej samej funkcji zwrotnej, w której zostało wcześniej wyrejestrowane, jest ono traktowane jako operacja zastąpienia; żadne wywołania początkowe lub końcowe nie są wywoływane.

Można użyć CBREG z CTLSU lub CTLRE.

CUNR
Zatrzymaj odbieranie komunikatów dla uchwytu obiektu i usuń uchwyt z tych, które zostały zakwalifikowane do wywołania zwrotnego.

Wywołanie zwrotne jest automatycznie wyrejestrowywane, jeśli powiązany uchwyt jest zamknięty.

Jeśli funkcja CBUNR jest wywoływana z poziomu konsumenta, a wywołanie zwrotne ma zdefiniowane wywołanie zatrzymania, jest ono wywoływane po powrocie od konsumenta.

Jeśli ta operacja zostanie wykonana dla *Hobj* bez zarejestrowanego konsumenta, wywołanie zostanie zwrócone z kodem powrotu RC2448.

CTLSU
Zawiesza odbieranie komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie mieć zdarzeń zawieszonych, a wszystkie zdarzenia pominięte w stanie zawieszenia nie będą udostępniane operacji po jej wznowieniu.

Po zawieszeniu funkcja konsumenta kontynuuje pobieranie wywołań zwrotnych typu sterującego.

STLRE
Wznawia odbieranie komunikatów dla uchwytu obiektu.

Jeśli ta operacja zostanie zastosowana do procedury obsługi zdarzeń, procedura obsługi zdarzeń nie będzie mieć zdarzeń zawieszonych, a wszystkie zdarzenia pominięte w stanie zawieszenia nie będą udostępniane operacji po jej wznowieniu.

CBDSC (MQCBD)-wejście

Zarządzanie funkcją zwrotną-parametr CBDSC.

Jest to struktura identyfikująca funkcję zwrotną, która jest rejestrowana przez aplikację, oraz opcje używane podczas jej rejestrowania.

Szczegółowe informacje na temat struktury zawiera sekcja "MQCBD-deskryptor wywołania zwrotnego" na stronie 293 .

Deskryptor wywołania zwrotnego jest wymagany tylko dla opcji CBREG; jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Zarządzanie funkcją zwrotną-parametr HOBJ.

Ten uchwyt reprezentuje dostęp, który został ustanowiony dla obiektu, z którego ma być pobierany komunikat. Jest to uchwyt zwrócony z poprzedniego wywołania MQOPEN lub MQSUB (w parametrze **HOBJ**).

Parametr *HOBJ* nie jest wymagany podczas definiowania procedury obsługi zdarzeń (CBTEH) i musi być określony jako HONONE.

Jeśli funkcja *Hobj* została zwrócona z wywołania MQOPEN, kolejka musi być otwarta z co najmniej jedną z następujących opcji:

- OOOINPS
- OOOINPX
- OOINPQ,
- OOBRW

MSGDSC (MQMD)-wejście

Zarządzanie funkcją zwrotną -parametr MSGDSC.

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu.

Parametr **MsgDesc** definiuje atrybuty komunikatów wymaganych przez konsumenta oraz wersję deskryptora MQMD przekazywanego do konsumenta komunikatów.

Do wyboru komunikatów używane są parametry *MsgId*, *CorrelId*, *GroupId*, *MsgSeqNumber*i *Offset* w strukturze MQMD, w zależności od opcji określonych w parametrze **GetMsgOpts** .

Wartości *Encoding* i *CodedCharSetId* są używane do konwersji komunikatów, jeśli zostanie podana opcja GMCONV.

Szczegółowe informacje na ten temat zawiera sekcja MQMD .

MsgDesc jest używana tylko dla CBREG i, jeśli wymagane są wartości inne niż domyślne dla dowolnych pól. Wartość *MsgDesc* nie jest używana dla procedury obsługi zdarzeń.

Jeśli deskryptor nie jest wymagany, przekazany adres parametru może mieć wartość NULL.

Należy zauważyć, że jeśli wiele konsumentów jest zarejestrowanych dla tej samej kolejki z nakładającymi się selektorami, wybrany konsument dla każdego komunikatu jest niezdefiniowany.

GMO (MQGMO)-wejście

Zarządzanie funkcją zwrotną-parametr GMO.

Opcje sterujące sposobem, w jaki konsument komunikatów otrzymuje komunikaty.

Wszystkie opcje mają znaczenie opisane w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106, jeśli są używane w wywołaniu MQGET, z wyjątkiem następujących:

GMSSIG
Ta opcja nie jest dozwolona.

GMBRWF, GMBRWN, GMMBH, GMMBC,
Kolejność komunikatów dostarczanych do konsumenta przeglądania zależy od kombinacji tych opcji. Istotne kombinacje to:

GMBRWF,
Pierwszy komunikat w kolejce jest wielokrotnie dostarczany do konsumenta. Jest to przydatne, gdy konsument niszczy komunikat w wywołaniu zwrotnym. Tej opcji należy używać ostrożnie.

GMBRWN
Konsument otrzymuje każdy komunikat w kolejce, od bieżącej pozycji kursora do osiągnięcia końca kolejki.

GMBRWF + GMBRWN
Kursor zostanie zresetowany do początku kolejki. Konsumentowi są następnie nadawane wszystkie komunikaty, dopóki kursor nie osiągnie końca kolejki.

GMBRWF + GMMBH lub GMMBC
Począwszy od początku kolejki, konsumentowi jest nadawany pierwszy nieoznaczony komunikat w kolejce, który jest następnie oznaczany dla tego konsumenta. Ta kombinacja zapewnia, że konsument może odbierać nowe komunikaty dodane za bieżącym punktem kursora.

GMBRWN + GMMBH lub GMMBC
Począwszy od pozycji kursora konsument otrzymuje następny nieoznaczony komunikat w kolejce, który jest następnie oznaczany dla tego konsumenta. Tej kombinacji należy używać ostrożnie, ponieważ komunikaty mogą być dodawane do kolejki za bieżącą pozycją kursora.

GMBRWF + GMBRWN + GMMBH lub GMMBC
Ta kombinacja nie jest dozwolona, jeśli zostanie użyta, wywołanie zwraca wartość RC2046.

GMNWT, GMWT i GMWI
Te opcje sterują sposobem wywoływania konsumenta.

GMNWT,
Konsument nigdy nie jest wywoływany z kodem powrotu RC2033. Konsument jest wywoływany tylko dla komunikatów i zdarzeń

GMWT z zerem GMWI
Kod RC2033 jest przekazywany do konsumenta tylko wtedy, gdy nie ma komunikatów i

- Konsument został uruchomiony
- Konsument został dostarczony co najmniej jeden komunikat od ostatniego kodu przyczyny braku komunikatu.

Zapobiega to odpytywaniu konsumenta w pętli zajętości, gdy określony jest zerowy odstęp czasu oczekiwania.

GMWT i dodatni GMWI
Użytkownik jest wywoływany po upływie określonego czasu oczekiwania z kodem przyczyny RC2033. To wywołanie jest wykonywane niezależnie od tego, czy jakiekolwiek komunikaty zostały dostarczone do konsumenta. Pozwala to użytkownikowi na wykonywanie przetwarzania pulsu lub przetwarzania wsadowego.

GMWT i GMWI WIULIM
Określa nieskończony czas oczekiwania przed zwróceniem wartości RC2033. Konsument nigdy nie jest wywoływany z kodem powrotu RC2033.

GMO jest używana tylko dla CBREG i, jeśli wymagane są wartości inne niż domyślne dla dowolnych pól. Wartość *GMO* nie jest używana dla procedury obsługi zdarzeń.

Jeśli opcje nie są wymagane, przekazany adres parametru może mieć wartość NULL.

Jeśli uchwyt właściwości komunikatu jest udostępniany w strukturze MQGMO, kopia jest udostępniana w strukturze MQGMO, która jest przekazywana do wywołania zwrotnego konsumenta. Po powrocie z wywołania MQCB aplikacja może usunąć uchwyt właściwości komunikatu.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Zarządzanie funkcją zwrotną-parametr CMPCOD.

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Zarządzaj funkcją zwrotną-parametr REASON.

Poniższe kody przyczyny są kodami, które menedżer kolejek może zwrócić dla parametru **REASON** .

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* to CCFAIL:

RC2204
(2204, X'89C') Adapter nie jest dostępny.

RC2133
(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2374
(2374, X' 946 ') Wyjście API nie powiodło się.

RC2183
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2005
(2005, X'7D5') Niepoprawny parametr długości buforu.

RC2219
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2487
(2487, X'9B7') Niepoprawne pole typu wywołania zwrotnego.

RC2448
(2448, X' 990 ') Nie można wyrejestrować, zawiesić lub wznowić, ponieważ nie ma zarejestrowanego wywołania zwrotnego.

RC2486
(2486, X'9B6') Należy podać wartość *CallbackFunction* lub *CallbackName* , ale nie obie jednocześnie.

RC2483
(2483, X'9B3') Niepoprawne pole typu wywołania zwrotnego.

RC2484
(2484, X'9B4') Niepoprawne pole opcji MQCBD.

RC2140
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2217
(2217, X'8A9') Brak uprawnień do połączenia.

RC2202
(2202, X'89A') wygaszanie połączenia.

RC2203
(2203, X'89B') Połączenie jest zamykane.

RC2207
(2207, X'89F') Błąd identyfikatora korelacji.

RC2010
(2010, X'7DA') Niepoprawny parametr długości danych.

RC2016
(2016, X'7E0') Liczba pobrań zablokowana dla kolejki.

RC2351
(2351, X'92F') Globalny konflikt jednostek pracy.

RC2186
(2186, X'88A') Niepoprawna struktura opcji Get-message.

RC2353
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2259
(2259, X'8D3') Niespójna specyfikacja przeglądania.

RC2245
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2246
(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

RC2352
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

RC2247
(2247, X'8C7') Opcje zgodności są niepoprawne.

RC2485
(2485, X'9B4') Niepoprawne pole *MaxMsgLength* .

RC2026
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

RC2497
(2497, X'9C1') W module nie można znaleźć określonego punktu wejścia funkcji.

RC2496
(2496, X'9C0') Znaleziono moduł, ale ma on niepoprawny typ; nie jest to 32-bitowa, 64-bitowa lub poprawna biblioteka dołączana dynamicznie.

RC2495
(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie ma uprawnień do ładowania.

RC2250
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

RC2331
(2331, X'91B') Użycie znacznika komunikatu jest niepoprawne.

RC2033
(2033, X'7F1') Brak dostępnego komunikatu.

RC2034
(2034, X'7F2') Kursor przeglądania nie jest ustawiony na komunikacie.

RC2036
(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

RC2037
(2037, X'7F5') Kolejka nie jest otwarta do wprowadzania.

RC2041
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2206
(2206, X'89E') Niepoprawny kod operacji w wywołaniu API.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2193
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2394
(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161
(2161, X'871 ') wygaszanie menedżera kolejek.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2069
(2069, X'815 ') Sygnał zaległości dla tego uchwytu.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2109
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

RC2024
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

RC2072
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2354
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

RC2355
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

RC2255
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

RC2090
(2090, X'82A') Niepoprawny odstęp czasu oczekiwania w MQGMO.

RC2256
(2256, X'8D0') Podano niewłaściwą wersję MQGMO.

RC2257
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

RC2298
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCB(HCONN : OPERATN : CBDSC :
                                HOBJ : MSGDSC : GMO :
                                DATLEN : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQCB             PR                  EXTPROC('MQCB')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Operation
D OPERATN                        10I 0 VALUE
D* Callback descriptor
D CBDSC                         180A
D* Object handle
D HOBJ                           10I 0 VALUE
D* Message Descriptor
D MSGDSC                        364A
D* Get options
D GMO                           112A
D* Completion code
D CMPCOD                         10I 0
* Reason code qualifying CompCode
D REASON                         10I 0
```

IBM i MQCLOSE (Close object-Zamknij obiekt) w systemie IBM i

Wywołanie MQCLOSE powoduje ponowne uzyskanie dostępu do obiektu i jest odwrotnością wywołania MQOPEN.

Składnia

MQCLOSE *(HCONN, HOBJ, OPTS, CMPCOD, REASON)*

Użycie notatek

1. Jeśli aplikacja wywoła wywołanie MQDISC lub zakończy działanie normalnie lub nieprawidłowo, wszystkie obiekty, które zostały otwarte przez aplikację i nadal są otwarte, są zamykane automatycznie z opcją CONONE.
2. Jeśli zamykany obiekt jest *kolejką*, mają zastosowanie następujące punkty:
   - Jeśli operacje w kolejce są wykonywane w ramach jednostki pracy, można zamknąć kolejkę przed lub po wystąpieniu punktu synchronizacji bez wpływu na wynik punktu synchronizacji.

- Jeśli kolejka została otwarta z opcją OOBRW, kursor przeglądania jest niszczony. Jeśli kolejka zostanie później ponownie otwarta z opcją OOBRW, zostanie utworzony nowy kursor przeglądania (patrz opis opcji OOBRW w MQOPEN).
- Jeśli komunikat jest aktualnie zablokowany dla tego uchwytu w czasie wywołania MQCLOSE, blokada jest zwalniana (patrz opcja GMLK opisana w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106).

3. Poniższe punkty mają zastosowanie, jeśli zamykany obiekt jest *kolejką dynamiczną* (trwałą lub tymczasową):
   - W przypadku kolejki dynamicznej można określić opcje CODEL lub COPURG niezależnie od opcji określonych w odpowiednim wywołaniu MQOPEN.
   - Po usunięciu kolejki dynamicznej wszystkie wywołania MQGET z opcją GMWT, które oczekują w kolejce, są anulowane i zwracany jest kod przyczyny RC2052 . Patrz opis opcji GMWT w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106.

     Po usunięciu kolejki dynamicznej każde wywołanie (inne niż MQCLOSE), które próbuje odwołać się do kolejki przy użyciu wcześniej uzyskanego uchwytu *HOBJ* , kończy się niepowodzeniem z kodem przyczyny RC2052.

     Należy pamiętać, że chociaż aplikacje nie mogą uzyskać dostępu do usuniętej kolejki, kolejka nie jest usuwana z systemu, a powiązane z nią zasoby nie są zwalniane, dopóki wszystkie uchwyty odwołujące się do kolejki nie zostaną zamknięte, a wszystkie jednostki pracy wpływające na kolejkę nie zostaną zatwierdzone lub wycofane.
   - Jeśli podczas usuwania trwałej kolejki dynamicznej, jeśli uchwyt *HOBJ* określony w wywołaniu MQCLOSE nie jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, zostanie wykonane sprawdzenie, czy identyfikator użytkownika, który był używany do sprawdzania poprawności wywołania MQOPEN, ma uprawnienia do usuwania kolejki. Jeśli w wywołaniu MQOPEN podano opcję OOALTU, sprawdzany identyfikator użytkownika to *ODAU*.

     To sprawdzenie nie jest wykonywane, jeśli:

     – Podany uchwyt jest zwracany przez wywołanie MQOPEN, które utworzyło kolejkę.

     – Usuwana kolejka jest tymczasową kolejką dynamiczną.
   - Po zamknięciu tymczasowej kolejki dynamicznej, jeśli uchwyt *HOBJ* określony w wywołaniu MQCLOSE jest tym, który został zwrócony przez wywołanie MQOPEN, które utworzyło kolejkę, zostanie ona usunięta. Dzieje się tak niezależnie od opcji zamykania określonych w wywołaniu MQCLOSE. Jeśli w kolejce znajdują się komunikaty, są one usuwane; nie są generowane żadne komunikaty raportu.

     Jeśli istnieją niezatwierdzone jednostki pracy, które mają wpływ na kolejkę, kolejka i jej komunikaty są nadal usuwane, ale nie powoduje to niepowodzenia jednostek pracy. Jednak, jak opisano wcześniej, zasoby powiązane z jednostkami pracy nie są zwalniane, dopóki żadna z jednostek pracy nie zostanie zatwierdzona lub wycofana.

4. Poniższe punkty mają zastosowanie, jeśli zamykany obiekt jest *listą dystrybucyjną*:
   - Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest CONONE; wywołanie kończy się niepowodzeniem z kodem przyczyny RC2046 lub RC2045 , jeśli podano inne opcje.
   - Po zamknięciu listy dystrybucyjnej indywidualne kody zakończenia i kody przyczyny nie są zwracane dla kolejek znajdujących się na liście-tylko parametry **CMPCOD** i **REASON** wywołania są dostępne do celów diagnostycznych.

     Jeśli wystąpi błąd podczas zamykania jednej z kolejek, menedżer kolejek kontynuuje przetwarzanie i próbuje zamknąć pozostałe kolejki na liście dystrybucyjnej. Parametry **CMPCOD** i **REASON** wywołania są następnie ustawiane w celu zwrócenia informacji opisujących niepowodzenie. Dlatego kod zakończenia może mieć wartość CCFAIL, nawet jeśli większość kolejek została pomyślnie zamknięta. Kolejka, w której wystąpił błąd, nie została zidentyfikowana.

     Jeśli w więcej niż jednej kolejce wystąpi awaria, nie jest ona zdefiniowana, która awaria jest zgłaszana w parametrach **CMPCOD** i **REASON** .

Parametry

Wywołanie MQCLOSE ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście/wyjście

Uchwyt obiektu.

Ten uchwyt reprezentuje obiekt, który jest zamykany. Obiekt może być dowolnego typu. Wartość *HOBJ* została zwrócona przez poprzednie wywołanie MQOPEN.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia ten parametr na wartość, która nie jest poprawnym uchwytem dla środowiska. Wartość ta jest następująca:

HUNUH
Uchwyt obiektu nie do użycia.

OPTS (10-cyfrowa liczba całkowita ze znakiem)-wejście

Opcje sterujące działaniem komendy MQCLOSE.

Parametr **OPTS** steruje sposobem zamykania obiektu. Tylko trwałe kolejki dynamiczne i subskrypcje mogą być zamykane w więcej niż jeden sposób. Trwałe kolejki dynamiczne można zachować lub usunąć. Są to kolejki z atrybutem **DefinitionType** o wartości QDPERM (patrz opis atrybutu **DefinitionType** w sekcji “Atrybuty kolejek” na stronie 1410). Opcje zamykania zostały podsumowane w tabeli znajdującej się w dalszej części tego tematu.

Trwałe subskrypcje można zachować lub usunąć. Są one tworzone za pomocą wywołania MQSUB z opcją SODUR.

Podczas zamykania uchwytu do zarządzanego miejsca docelowego (czyli parametru **Hobj** zwróconego w wywołaniu MQSUB z opcją SOMAN) menedżer kolejek wyczyści wszystkie niepobrane publikacje, jeśli powiązana subskrypcja również została usunięta. W tym celu należy użyć opcji CORMSB w parametrze **Hsub** zwróconym w wywołaniu MQSUB. Należy zauważyć, że mechanizm CORMSB jest domyślnym zachowaniem programu MQCLOSE dla subskrypcji nietrwałej.

Podczas zamykania uchwytu do niezarządzanego miejsca docelowego użytkownik jest odpowiedzialny za czyszczenie kolejki, do której są wysyłane publikacje. Zaleca się najpierw zamknięcie subskrypcji przy użyciu mechanizmu CORMSB, a następnie przetworzenie komunikatów poza kolejką do momentu, gdy nie będzie już żadnych komunikatów.

Należy określić jeden (i tylko jeden) z następujących elementów:

Opcje zamykania kolejek dynamicznych

Następujące opcje sterują sposobem zamykania trwałych kolejek dynamicznych:

KOD
Usuń kolejkę.

Kolejka jest usuwana, jeśli spełniony jest jeden z następujących warunków:

- Jest to stała kolejka dynamiczna utworzona przez poprzednie wywołanie MQOPEN i nie ma w niej żadnych komunikatów ani niezatwierdzonych żądań pobierania lub umieszczania oczekujących dla kolejki (dla bieżącego zadania lub dowolnego innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło kod *HOBJ*. W takim przypadku wszystkie komunikaty w kolejce są czyszczone.

We wszystkich innych przypadkach, w tym w przypadku zwrócenia wartości *Hobj* w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2045, a obiekt nie jest usuwany.

KPOURG
Usuń kolejkę, usuwając wszystkie znajdujące się w niej komunikaty.

Kolejka jest usuwana, jeśli spełniony jest jeden z następujących warunków:

- Jest to trwała kolejka dynamiczna utworzona przez poprzednie wywołanie MQOPEN i nie ma niezatwierdzonych żądań pobrania lub umieszczenia oczekujących dla kolejki (dla bieżącego zadania lub dowolnego innego zadania).
- Jest to tymczasowa kolejka dynamiczna, która została utworzona przez wywołanie MQOPEN, które zwróciło kod *H0BJ*.

We wszystkich innych przypadkach, w tym w przypadku zwrócenia wartości *Hobj* w wywołaniu MQSUB, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2045, a obiekt nie jest usuwany.

W następnej tabeli przedstawiono poprawne opcje zamykania oraz informacje o tym, czy obiekt został zachowany, czy usunięty.

Tabela 743. Poprawne opcje zamykania do użycia z zachowanymi lub usuniętymi obiektami

Typ obiektu lub kolejki	**BRAK KONIE**	**KOD**	**KPOURG**
Obiekt inny niż kolejka	Zachowany	Niepoprawne	Niepoprawne
Predefiniowana kolejka	Zachowany	Niepoprawne	Niepoprawne
stała kolejka dynamiczna	Zachowany	Usunięte, jeśli puste i bez oczekujących aktualizacji	Komunikaty usunięte; kolejka usunięta, jeśli nie ma oczekujących aktualizacji
Tymczasowa kolejka dynamiczna (wywołanie wysłane przez twórcę kolejki)	Usunięte	Usunięte	Usunięte
Tymczasowa kolejka dynamiczna (wywołanie nie zostało wykonane przez twórcę kolejki)	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna	Zachowany	Niepoprawne	Niepoprawne
Miejsce docelowe subskrypcji zarządzanej	Zachowany	Niepoprawne	Niepoprawne
Lista dystrybucyjna (subskrypcja została usunięta)	Komunikaty usunięte; kolejka usunięta	Niepoprawne	Niepoprawne

Opcje zamykania subskrypcji

Te opcje określają, czy trwałe subskrypcje są usuwane po zamknięciu uchwytu oraz czy publikacje oczekujące na odczyt przez aplikację są czyszczone. Te opcje są poprawne tylko w przypadku użycia z uchwytem obiektu zwróconym w parametrze **HSUB** wywołania MQSUB.

KKPSB
Uchwyt subskrypcji jest zamknięty, ale wykonana subskrypcja jest zachowana. Publikacje będą nadal wysyłane do miejsca docelowego określonego w subskrypcji. Ta opcja jest poprawna tylko wtedy, gdy subskrypcja została wykonana z opcją SODUR. COKPSB jest wartością domyślną, jeśli subskrypcja jest trwała

CORMSB
Subskrypcja zostanie usunięta, a uchwyt do subskrypcji zostanie zamknięty.

Parametr **Hobj** wywołania MQSUB nie jest unieważniany przez zamknięcie parametru **Hsub** i może być nadal używany do odbierania pozostałych publikacji przez wywołania MQGET lub MQCB.

Po zamknięciu parametru **Hobj** wywołania MQSUB, jeśli było to zarządzane miejsce docelowe, wszystkie niepobrane publikacje zostaną usunięte.

Jeśli subskrypcja nie jest trwała, wartością domyślną jest CORMSB.

Te opcje zamykania subskrypcji zostały podsumowane w następujących tabelach:

Aby zamknąć uchwyt trwałej subskrypcji, ale pozostawić subskrypcję, należy użyć następujących opcji zamykania subskrypcji:

Tabela 744. Opcje zadania służące do zamykania uchwytu trwałej subskrypcji i opuszczania subskrypcji

Czynność	**Opcja zamknięcia subskrypcji**
Zachowaj publikacje w uchwycie MQOPENed	KKPSB
Usuń publikacje z uchwytu MQOPENed	Działanie niedozwolone
Przechowuj publikacje na uchwycie za pomocą SOMAN	KKPSB
Usuwanie publikacji na uchwycie za pomocą SOMAN	Działanie niedozwolone

Aby anulować subskrypcję, zamykając uchwyt subskrypcji trwałej i anulując jego subskrypcję lub zamykając uchwyt subskrypcji nietrwałej, należy użyć następujących opcji zamykania subskrypcji:

Tabela 745. Opcje zadania anulowania subskrypcji

Czynność	**Opcja zamknięcia subskrypcji**
Zachowaj publikacje w uchwycie MQOPENed	CORMSB
Usuń publikacje z uchwytu MQOPENed	Działanie niedozwolone
Przechowuj publikacje na uchwycie za pomocą SOMAN	CORMSB
Usuwanie publikacji na uchwycie za pomocą SOMAN	COPGSB

Opcje odczytu z wyprzedzeniem

Następujące opcje sterują tym, co dzieje się z nietrwałymi komunikatami, które zostały wysłane do klienta przed zażądaniem ich przez aplikację i nie zostały jeszcze wykorzystane przez aplikację. Te komunikaty są przechowywane w buforze odczytu z wyprzedzeniem klienta, który oczekuje na żądanie aplikacji i może zostać usunięty lub skonsumowany z kolejki przed zakończeniem operacji MQCLOSE.

CIMM
Obiekt jest zamykany natychmiast, a wszystkie komunikaty, które zostały wysłane do klienta przed zażądaniem ich przez aplikację, są odrzucane i nie są dostępne do wykorzystania przez żadną aplikację. Jest to wartość domyślna.

COQSC,
Żądanie zamknięcia obiektu jest wykonywane, ale jeśli jakiekolwiek komunikaty, które zostały wysłane do klienta przed zażądaniem ich przez aplikację, nadal znajdują się w buforze odczytu z wyprzedzeniem klienta, wywołanie MQCLOSE zwróci kod ostrzeżenia RC2458, a uchwyt obiektu pozostanie poprawny.

Aplikacja może kontynuować używanie uchwytu obiektu do pobierania komunikatów, dopóki nie będzie więcej dostępnych, a następnie ponownie zamknąć obiekt. Więcej komunikatów nie będzie wysyłanych do klienta przed aplikacją żądającą, odczyt z wyprzedzeniem jest teraz wyłączony.

Zaleca się, aby aplikacje używały komendy COQSC zamiast próby osiągnięcia punktu, w którym nie ma więcej komunikatów w buforze odczytu z wyprzedzeniem klienta, ponieważ komunikat może nadejść między ostatnim wywołaniem MQGET i następującym po nim wywołaniem MQCLOSE, który zostałby odrzucony w przypadku użycia komendy COIMM.

Jeśli komenda MQCLOSE z COQSC jest wykonywana z poziomu asynchronicznej funkcji zwrotnej, stosowane jest to samo zachowanie odczytu komunikatów z wyprzedzeniem. Jeśli zostanie zwrócony kod ostrzeżenia RC2458 , funkcja zwrotna zostanie wywołana co najmniej raz jeszcze. Gdy ostatni komunikat, który został odczytany z wyprzedzeniem, został przekazany do funkcji zwrotnej, pole CBCFLG jest ustawione na CBCFBE.

Opcja domyślna

Jeśli nie jest wymagana żadna z opisanych wcześniej opcji, można użyć następującej opcji:

BRAK KONIE
Nie jest wymagane opcjonalne przetwarzanie zamknięcia.

Należy to określić dla:

- Obiekty inne niż kolejki
- Kolejki predefiniowane
- Tymczasowe kolejki dynamiczne (ale tylko w przypadkach, gdy parametr *HOBJ* nie jest uchwytem zwróconym przez wywołanie MQOPEN, które utworzyło kolejkę).
- Lista dystrybucyjna

We wszystkich poprzednich przypadkach obiekt jest zachowywany i nie jest usuwany.

Jeśli ta opcja jest określona dla tymczasowej kolejki dynamicznej:

- Kolejka zostanie usunięta, jeśli została utworzona przez wywołanie MQOPEN, które zwróciło kod *HOBJ* . wszystkie komunikaty znajdujące się w kolejce są czyszczone.
- We wszystkich innych przypadkach kolejka (i wszystkie znajdujące się na niej komunikaty) są zachowywane.

Jeśli ta opcja jest określona dla trwałej kolejki dynamicznej, kolejka jest zachowywana i nie jest usuwana.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2241
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

RC2242
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

Jeśli *CMPCOD* to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2035
(2035, X'7F3') Brak uprawnień dostępu.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2045
(2045, X'7FD') Opcja niepoprawna dla typu obiektu.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2055
(2055, X'807 ') Kolejka zawiera jeden lub więcej komunikatów lub niezatwierdzone żądania umieszczania lub pobierania.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2063
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCLOSE(HCONN : HOBJ : OPTS :
C                                       CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCLOSE          PR                  EXTPROC('MQCLOSE')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Object handle
D HOBJ                          10I 0
D* Options that control the action of MQCLOSE
D OPTS                          10I 0 VALUE
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQCMIT (Zatwierdzanie zmian) w systemie IBM i

Wywołanie MQCMIT wskazuje menedżerowi kolejek, że aplikacja osiągnęła punkt synchronizacji oraz że wszystkie operacje pobierania i umieszczania komunikatów, które wystąpiły od ostatniego punktu synchronizacji, mają być trwałe. Komunikaty umieszczone jako część jednostki pracy są udostępniane innym aplikacjom; komunikaty pobrane jako część jednostki pracy są usuwane.

Składnia

Komenda MQCMIT *(HCONN, COMCOD, REASON)*

Użycie notatek

Podczas korzystania z MQCMIT należy wziąć pod uwagę następujące uwagi dotyczące składni.

1. Tego wywołania można użyć tylko wtedy, gdy sam menedżer kolejek koordynuje jednostkę pracy. Jest to lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiedniego wywołania zatwierdzenia zamiast MQCMIT. Środowisko może również obsługiwać niejawne zatwierdzanie spowodowane przez normalne zakończenie działania aplikacji.
   - W systemie IBM itego wywołania można użyć dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie została wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji "MQDISC (Rozłączenie menedżera kolejek) w systemie IBM i" na stronie 1330 .
4. Gdy aplikacja umieszcza lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Te informacje są powiązane z uchwytem kolejki i obejmują takie elementy, jak:
   - Wartości pól *MDGID*, *MDSEQ*, *MDOFF*i *MDMFL* w strukturze MQMD.
   - Określa, czy komunikat jest częścią jednostki pracy.
   - Dla wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

   Po zatwierdzeniu jednostki pracy menedżer kolejek zachowuje informacje o grupie i segmencie, a aplikacja może kontynuować umieszczanie lub pobieranie komunikatów w bieżącej grupie komunikatów lub w komunikacie logicznym.

   Zachowanie informacji o grupie i segmencie po zatwierdzeniu jednostki pracy umożliwia aplikacji rozmieszczenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów w kilku jednostkach pracy. Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną pamięć kolejki. Jednak aplikacja musi zachować wystarczającą ilość informacji, aby w przypadku awarii systemu możliwe było zrestartowanie umieszczania lub pobierania komunikatów w odpowiednim miejscu. Szczegółowe informacje na temat restartowania w odpowiednim punkcie po awarii systemu zawiera opis opcji PMLOGO w sekcji "MQPMO (opcje umieszczania komunikatów) w systemie IBM i" na stronie 1208oraz opis opcji GMLOGO w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106.

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

1. Jednostka pracy ma taki sam zasięg jak uchwyt połączenia. Oznacza to, że wszystkie wywołania IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wykonane przy użyciu innego uchwytu połączenia (na przykład wywołania wykonane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączeń zawiera opis parametru **HCONN** w sekcji MQCONN.
2. To wywołanie ma wpływ tylko na komunikaty, które zostały umieszczone lub pobrane jako część bieżącej jednostki pracy.
3. Aplikacja długotrwała, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w ramach jednostki pracy, ale która nigdy nie wysyła wywołania zatwierdzenia lub wywołania powrotnego, może spowodować zapełnienie kolejek komunikatami, które nie są dostępne dla innych aplikacji. W celu zabezpieczenia się przed taką możliwością administrator powinien ustawić atrybut menedżera kolejek **MaxUncommittedMsgs** na wartość, która jest na tyle niska, aby zapobiec zapełnianiu kolejek przez aplikacje, ale na tyle wysoka, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQCMIT ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

COMCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *COMCOD*.

Jeśli *COMCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *COMCOD* ma wartość CCWARN:

RC2003
(2003, X'7D3') Jednostka pracy wycofała się.

RC2124
(2124, X'84C') Wynik operacji zatwierdzania jest w toku.

Jeśli *COMCOD* to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2123
(2123, X'84B') Wynik operacji zatwierdzania lub wycofania jest mieszany.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQCMIT(HCONN : COMCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCMIT           PR                  EXTPROC('MQCMIT')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Completion code
D COMCOD                        10I 0
D* Reason code qualifying COMCOD
D REASON                        10I 0
```

IBM i MQCONN (połączenie z menedżerem kolejek) w systemie IBM i

Wywołanie MQCONN łączy aplikację z menedżerem kolejek. Udostępnia on uchwyt połączenia menedżera kolejek, który jest używany przez aplikację w kolejnych wywołaniach kolejkowania komunikatów.

- Aplikacje muszą używać wywołania MQCONN lub MQCONNX w celu nawiązania połączenia z menedżerem kolejek, a wywołania MQDISC w celu rozłączenia z menedżerem kolejek.

W systemie IBM MQ for Multiplatformskażdy wątek w aplikacji może łączyć się z różnymi menedżerami kolejek. W innych systemach wszystkie współbieżne połączenia w procesie muszą być nawiązywane z tym samym menedżerem kolejek.

- "Składnia" na stronie 1316
- "Użycie notatek" na stronie 1317
- "Parametry" na stronie 1317
- "Deklaracja RPG" na stronie 1320

Składnia

(QMNAME, HCONN, CMPCOD, REASON) MQCONN

Użycie notatek

1. Menedżer kolejek, z którym nawiązywane jest połączenie przy użyciu wywołania MQCONN, jest nazywany *lokalnym menedżerem kolejek*.
2. Kolejki, których właścicielem jest menedżer kolejek lokalnych, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie z nich komunikatów.

   Kolejki współużytkowane, których właścicielem jest grupa współużytkowania kolejek, do której należy menedżer kolejek lokalnych, są wyświetlane w aplikacji jako kolejki lokalne. Możliwe jest umieszczanie komunikatów w tych kolejkach i pobieranie z nich komunikatów.

   Kolejki należące do menedżerów kolejek zdalnych są wyświetlane jako kolejki zdalne. Możliwe jest umieszczanie komunikatów w tych kolejkach, ale nie jest możliwe pobieranie komunikatów z tych kolejek.
3. Jeśli działanie menedżera kolejek zakończy się niepowodzeniem podczas działania aplikacji, aplikacja musi ponownie wywołać komendę MQCONN, aby uzyskać nowy uchwyt połączenia do użycia w kolejnych wywołaniach programu IBM MQ . Aplikacja może okresowo wywoływać wywołanie MQCONN, dopóki wywołanie nie zakończy się powodzeniem.

   Jeśli aplikacja nie jest pewna, czy jest połączona z menedżerem kolejek, może bezpiecznie wywołać wywołanie MQCONN w celu uzyskania uchwytu połączenia. Jeśli aplikacja jest już połączona, zwrócony uchwyt jest taki sam, jak zwrócony przez poprzednie wywołanie MQCONN, ale z kodem zakończenia CCWARN i kodem przyczyny RC2002.
4. Po zakończeniu używania przez aplikację wywołań IBM MQ aplikacja powinna użyć wywołania MQDISC w celu rozłączenia się z menedżerem kolejek.
5. W systemie IBM iprogramy zakończone nieprawidłowo nie są automatycznie odłączane od menedżera kolejek. Dlatego aplikacje powinny być napisane w celu umożliwienia wywołania MQCONN lub MQCONNX zwracającego kod zakończenia CCWARN i kod przyczyny RC2002. Uchwyt połączenia zwrócony w tej sytuacji może być używany normalnie.

Parametry

Wywołanie MQCONN ma następujące parametry:

QMNAME (48-bajtowy łańcuch znaków)-wejście

Nazwa menedżera kolejek.

Jest to nazwa menedżera kolejek, z którym aplikacja chce się połączyć. Nazwa może zawierać następujące znaki:

- Wielkie litery (od A do Z)
- Małe litery (od a do z)
- Cyfry (od 0 do 9)
- Kropka (.), ukośnik (/), podkreślenie (_), procent (%)

Nazwa nie może zawierać początkowych ani wewnętrznych odstępów, ale może zawierać końcowe odstępy. Znak o kodzie zero może być używany do wskazania końca istotnych danych w nazwie; znak o kodzie zero i wszystkie następujące po nim znaki są traktowane jako odstępy. W podanych środowiskach obowiązują następujące ograniczenia:

- W systemie IBM inazwy zawierające małe litery, ukośnik lub procent muszą być ujęte w cudzysłów, jeśli zostały podane w komendach. Te cudzysłowy nie mogą być podawane w parametrze **QMNAME** .

Jeśli nazwa składa się wyłącznie z odstępów, używana jest nazwa *domyślnego* menedżera kolejek.

Nazwa określona dla parametru *QMNAME* musi być nazwą *możliwego do połączenia* menedżera kolejek.

Grupy współużytkowania kolejek:W systemach, w których istnieje kilka menedżerów kolejek skonfigurowanych w celu utworzenia grupy współużytkowania kolejek, nazwę grupy współużytkowania kolejek można określić dla parametru *QMNAME* zamiast nazwy menedżera kolejek. Umożliwia to aplikacji nawiązanie połączenia z *dowolnym* menedżerem kolejek, który jest dostępny

w grupie współużytkowania kolejek. System można również skonfigurować w taki sposób, aby pusta wartość *QMNAME* powodująca połączenie z grupą współużytkowania kolejek zamiast z domyślnym menedżerem kolejek.

Jeśli parametr *QMNAME* określa nazwę grupy współużytkowania kolejek, ale w systemie istnieje również menedżer kolejek o tej nazwie, nawiązywane jest połączenie z tym drugim menedżerem kolejek, a nie z tym pierwszym. Tylko wtedy, gdy nawiązanie połączenia nie powiedzie się, zostanie podjęta próba nawiązania połączenia z jednym z menedżerów kolejek w grupie współużytkowania kolejek.

Jeśli połączenie zostanie pomyślnie nawiązane, uchwyt zwrócony przez wywołanie MQCONN lub MQCONNX może zostać użyty w celu uzyskania dostępu do *wszystkich* zasobów (zarówno współużytkowanych, jak i niewspółużytkowanych) należących do konkretnego menedżera kolejek, z którym nawiązano połączenie. Dostęp do tych zasobów podlega typowym kontrolom autoryzacji.

Jeśli aplikacja wysyła dwa wywołania MQCONN lub MQCONNX w celu nawiązania połączeń współbieżnych, a jedno lub oba wywołania określają nazwę grupy współużytkowania kolejek, drugie wywołanie może zwrócić kod zakończenia CCWARN i kod przyczyny RC2002. Taka sytuacja ma miejsce, gdy drugie wywołanie łączy się z tym samym menedżerem kolejek, co pierwsze wywołanie.

Grupy współużytkowania kolejek są obsługiwane tylko w systemie z/OS. Połączenie z grupą współużytkowania kolejek jest obsługiwane tylko w środowiskach wsadowych, RRS i TSO.

Aplikacje klienckie IBM MQ: w przypadku aplikacji IBM MQ MQI client podejmowana jest próba nawiązania połączenia dla każdej definicji kanału połączenia klienckiego o określonej nazwie menedżera kolejek, dopóki połączenie nie zostanie pomyślnie nawiązane. Jednak menedżer kolejek musi mieć taką samą nazwę, jak określona nazwa. Jeśli zostanie podana pusta nazwa, podejmowana jest próba nawiązania połączenia z każdym kanałem połączenia klienckiego z pustą nazwą menedżera kolejek do momentu pomyślnego zakończenia. W takim przypadku nie jest sprawdzana rzeczywista nazwa menedżera kolejek.

IBM MQ Grupy menedżerów kolejek klienckich: Jeśli podana nazwa rozpoczyna się od gwiazdki (*), rzeczywisty menedżer kolejek, z którym nawiązywane jest połączenie, może mieć nazwę inną niż nazwa określona przez aplikację. Podana nazwa (bez gwiazdki) definiuje *grupę* menedżerów kolejek zakwalifikowanych do połączenia. Implementacja wybiera jedną z grupy, próbując kolejno każdą z nich, w porządku alfabetycznym, aż do znalezienia jednej z nich, z którą można nawiązać połączenie. Jeśli żaden z menedżerów kolejek w grupie nie jest dostępny dla połączenia, wywołanie nie powiedzie się. Każdy menedżer kolejek jest wypróbowany tylko raz. Jeśli dla nazwy podano samą gwiazdkę, zostanie użyta domyślna grupa menedżerów kolejek zdefiniowana przez implementację.

Grupy menedżerów kolejek są obsługiwane tylko w przypadku aplikacji działających w środowisku klienta produktu MQ. Wywołanie nie powiedzie się, jeśli aplikacja niekliencka określi nazwę menedżera kolejek rozpoczynającą się od gwiazdki. Grupa jest definiowana przez udostępnienie kilku definicji kanału połączenia klienckiego o tej samej nazwie menedżera kolejek (określonej nazwie bez gwiazdki) w celu komunikowania się z każdym menedżerem kolejek w grupie. Grupa domyślna jest definiowana przez podanie co najmniej jednej definicji kanału połączenia klienckiego, z której każda ma pustą nazwę menedżera kolejek (podanie nazwy all-blank ma taki sam skutek, jak podanie pojedynczej gwiazdki dla nazwy aplikacji klienckiej).

Po nawiązaniu połączenia z jednym menedżerem kolejek grupy aplikacja może określić odstępy w typowy sposób w polach nazw menedżera kolejek w komunikacie i deskryptorach obiektów, co oznacza nazwę menedżera kolejek, z którym aplikacja rzeczywiście nawiązała połączenie (*menedżer kolejek lokalnych*). Jeśli aplikacja musi znać tę nazwę, można wydać wywołanie MQINQ w celu uzyskania informacji o atrybucie menedżera kolejek **QMgrName** .

Poprzedzenie nazwy połączenia gwiazdką oznacza, że aplikacja nie jest zależna od połączenia z konkretnym menedżerem kolejek w grupie. Odpowiednie zastosowania to:

- Aplikacje, które umieszczają komunikaty, ale ich nie dostają.
- Aplikacje, które umieszczają komunikaty żądań, a następnie pobierają komunikaty odpowiedzi z *tymczasowej kolejki dynamicznej* .

Nieodpowiednie aplikacje to aplikacje, które muszą pobrać komunikaty z określonej kolejki w danym menedżerze kolejek. Takie aplikacje nie powinny poprzedzać nazwy gwiazdką.

Należy zauważyć, że jeśli zostanie podana gwiazdka, maksymalna długość pozostałej części nazwy wynosi 47 znaków.

Długość tego parametru jest określona przez LNQMN.

HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Musi być ona określona we wszystkich kolejnych wywołaniach kolejkowania komunikatów wysyłanych przez aplikację. Przestaje być poprawna, gdy zostanie wydane wywołanie MQDISC lub gdy jednostka przetwarzania definiująca zasięg uchwytu zostanie zakończona.

Zakres uchwytu jest ograniczony do najmniejszej jednostki Przetwarzanie równoległe obsługiwane przez platformę, na której działa aplikacja. Uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołano wywołanie MQCONN.

- W systemie IBM izasięgiem uchwytu jest zadanie wywołujące wywołanie.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2002
(2002, X'7D2') Aplikacja jest już podłączona.

Jeśli *CMPCOD* to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2267
(2267, X'8DB') Nie można załadować wyjścia obciążenia klastra.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2035
(2035, X'7F3') Brak uprawnień dostępu.

RC2137
(2137, X'859 ') Obiekt nie został pomyślnie otwarty.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161
(2161, X'871 ') wygaszanie menedżera kolejek.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2063
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                    CALLP     MQCONN(QMNAME : HCONN : CMPCOD :
C                                      REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCONN           PR                  EXTPROC('MQCONN')
D* Name of queue manager
D QMNAME                        48A
D* Connection handle
D HCONN                         10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQCONNX (połączenie z menedżerem kolejek (rozszerzone)) w systemie IBM i

Wywołanie MQCONNX łączy aplikację z menedżerem kolejek. Udostępnia on uchwyt połączenia menedżera kolejek, który jest używany przez aplikację w kolejnych wywołaniach programu IBM MQ .

Wywołanie MQCONNX jest podobne do wywołania MQCONN, z tą różnicą, że MQCONNX umożliwia określenie opcji sterujących sposobem działania wywołania.

W systemie IBM MQ for Multiplatformskażdy wątek w aplikacji może łączyć się z różnymi menedżerami kolejek. W innych systemach wszystkie współbieżne połączenia w procesie muszą być nawiązywane z tym samym menedżerem kolejek.

- “Składnia” na stronie 1320
- “Parametry” na stronie 1321
- “Deklaracja RPG” na stronie 1321

Składnia

(QMNAME, CNOPT, HCONN, CMPCOD, REASON) MQCONNX

Parametry

Wywołanie MQCONNX ma następujące parametry:

QMNAME (48-bajtowy łańcuch znaków)-wejście

Nazwa menedżera kolejek.

Szczegółowe informacje zawiera opis parametru **QMNAME** w sekcji “MQCONN (połączenie z menedżerem kolejek) w systemie IBM i” na stronie 1316 .

CNOPT (MQCNO)-wejście/wyjście

Opcje sterujące działaniem komendy MQCONNX.

Szczegółowe informacje można znaleźć w sekcji “MQCNO (opcje połączenia) w systemie IBM i” na stronie 1076.

HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Uchwyt połączenia.

Szczegółowe informacje zawiera opis parametru **HCONN** w sekcji “MQCONN (połączenie z menedżerem kolejek) w systemie IBM i” na stronie 1316 .

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Szczegółowe informacje zawiera opis parametru **CMPCOD** w sekcji “MQCONN (połączenie z menedżerem kolejek) w systemie IBM i” na stronie 1316 .

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Szczegółowe informacje na temat możliwych kodów przyczyny zawiera opis parametru **REASON** w sekcji “MQCONN (połączenie z menedżerem kolejek) w systemie IBM i” na stronie 1316 .

Wywołanie MQCONNX może zwrócić następujące dodatkowe kody przyczyny:

Jeśli *CMPCOD* to CCFAIL:

RC2278
(2278, X'8E6') Niepoprawne pola połączenia klienta.

RC2139
(2139, X'85B') Niepoprawna struktura opcji połączenia.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCONN(QMNAME : HCONN : CMPCOD :
C                                      REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQCONN           PR                  EXTPROC('MQCONN')
D* Name of queue manager
D QMNAME                         48A
D* Options that control the action of MQCONNX
D HCONN                         224A
D* Connection handle
D HCONN                          10I 0
D* Completion code
D CMPCOD                         10I 0
```

```
     D* Reason code qualifying CMPCOD
     D REASON                        10I 0
```

IBM i MQCRTMH (Tworzenie uchwytu komunikatu-Create message handle) w systemie IBM i

Wywołanie MQCRTMH zwraca uchwyt komunikatu.

Aplikacja może jej używać w kolejnych wywołaniach kolejkowania komunikatów:

- Użyj wywołania MQSETMP , aby ustawić właściwość uchwytu komunikatu.
- Za pomocą wywołania MQINQMP można uzyskać informacje o wartości właściwości uchwytu komunikatu.
- Użyj wywołania MQDLTMP , aby usunąć właściwość uchwytu komunikatu.

Uchwyt komunikatu może być używany w wywołaniach MQPUT i MQPUT1 w celu powiązania właściwości uchwytu komunikatu z właściwościami umieszczanego komunikatu. Podobnie, określając uchwyt komunikatu w wywołaniu MQGET, można uzyskać dostęp do właściwości pobieranego komunikatu przy użyciu uchwytu komunikatu po zakończeniu wywołania MQGET.

Użyj komendy MQDLTMH , aby usunąć uchwyt komunikatu.

- "Składnia" na stronie 1322
- "Parametry" na stronie 1322
- "Deklaracja RPG" na stronie 1324

Składnia

(Hconn, CrtMsgHOpts, Hmsg, CompCode, Reason) MQCRTMH

Parametry

Wywołanie MQCRTMH ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX. Jeśli połączenie z menedżerem kolejek przestaje być poprawne, a żadne wywołanie IBM MQ nie działa na uchwycie komunikatu, wywołanie MQDLTMH jest niejawnie wywoływane w celu usunięcia komunikatu.

Alternatywnie można podać następującą wartość:

HUNAS
Uchwyt połączenia nie reprezentuje połączenia z żadnym konkretnym menedżerem kolejek.

Jeśli ta wartość jest używana, uchwyt komunikatu musi zostać usunięty za pomocą jawnego wywołania komendy MQDLTMH , aby zwolnić przydzieloną do niego pamięć. Program IBM MQ nigdy niejawnie nie usuwa uchwytu komunikatu.

Musi istnieć co najmniej jedno poprawne połączenie z menedżerem kolejek ustanowione w wątku tworzącą uchwyt komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2018.

CRTOPT (MQCMHO)-wejście

Opcje sterujące działaniem komendy MQCRTMH. Szczegółowe informacje na ten temat zawiera sekcja MQCMHO .

HMSG (20-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Na wyjściu zwracany jest uchwyt komunikatu, który może być używany do ustawiania, sprawdzania i usuwania właściwości uchwytu komunikatu. Początkowo uchwyt komunikatu nie zawiera żadnych właściwości.

Z uchwytem komunikatu powiązany jest również deskryptor komunikatu. Początkowo ten deskryptor komunikatu zawiera wartości domyślne. Wartości powiązanych pól deskryptora komunikatu można ustawić i wysłać zapytanie za pomocą wywołań MQSETMP i MQINQMP. Wywołanie MQDLTMP resetuje pole deskryptora komunikatu do wartości domyślnej.

Jeśli parametr *HCONN* jest określony jako wartość HCUNAS, zwrócony uchwyt komunikatu może być używany w wywołaniach MQGET, MQPUT lub MQPUT1 z dowolnym połączeniem w jednostce przetwarzania, ale może być używany tylko przez jedno wywołanie IBM MQ w danym momencie. Jeśli uchwyt jest używany, gdy drugie wywołanie IBM MQ próbuje użyć tego samego uchwytu komunikatu, drugie wywołanie IBM MQ kończy się niepowodzeniem z kodem przyczyny RC2499.

Jeśli parametr *HCONN* nie ma wartości HCUNAS, zwrócony uchwyt komunikatu może być używany tylko w określonym połączeniu.

Ta sama wartość parametru *HCONN* musi być używana w kolejnych wywołaniach MQI, w których używany jest ten uchwyt komunikatu:

- MQDLTMH
- Komenda MQSETMP
- MQINQMP
- Komenda MQDLTMP
- MQMHBUF
- MQBUFMH

Zwrócony uchwyt komunikatu przestaje być poprawny, gdy dla uchwytu komunikatu zostanie wydane wywołanie MQDLTMH lub gdy jednostka przetwarzania definiująca zasięg uchwytu zostanie zakończona. Komenda MQDLTMH jest wywoływana niejawnie, jeśli podczas tworzenia uchwytu komunikatu podano konkretne połączenie, a połączenie z menedżerem kolejek przestaje być poprawne, na przykład w przypadku wywołania MQDBC.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2461
(2461, X'099D') Struktura opcji tworzenia uchwytu komunikatu jest niepoprawna.

RC2273
(2273, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2017
(2017, X'07E1') Brak dostępnych uchwytów.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2460
(2460, X'099C') Wskaźnik uchwytu komunikatu jest niepoprawny.

RC2046
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat zawiera sekcja “Kody powrotu dla IBM i (ILE RPG)” na stronie 1471.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCRTMH(HCONN : CRTOPT : HMSG :
                                        CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQCRTMH          PR                  EXTPROC('MQCRTMH')
D* Connection handle
D HCONN                           10I 0 VALUE
D* Options that control the action of MQCRTMH
D CRTOPT                          12A
D* Message handle
D HMSG                            20I 0
D* Completion code
D CMPCOD                          10I 0
D* Reason code qualifying CompCode
D REASON                          10I 0
```

IBM i MQCTL (sterujące wywołanie zwrotne) w systemie IBM i

Wywołanie MQCTL wykonuje działania sterujące uchwytami obiektu otwartymi dla połączenia.

- “Składnia” na stronie 1324
- “Użycie notatek” na stronie 1324
- “Parametry” na stronie 1325
- “Deklaracja RPG” na stronie 1329

Składnia

(Hconn, Operation, ControlOpts, CompCode, Reason) MQCTL

Użycie notatek

1. Procedury zwrotne muszą sprawdzać odpowiedzi ze wszystkich usług, które wywołują, a jeśli podprogram wykryje warunek, którego nie można rozstrzygnąć, musi wydać komendę MQCB (CBREG), aby zapobiec powtarzającym się wywołaniom procedury zwrotnej.

Parametry

Wywołanie MQCTL ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

OPERATN (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Operacja przetwarzana w wywołaniu zwrotnym zdefiniowanym dla określonego uchwytu obiektu. Należy podać jedną i tylko jedną z następujących opcji:

CLSR
Rozpocznij odbieranie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Wywołania zwrotne działają w wątku uruchomionym przez system, który jest inny niż każdy wątek aplikacji.

Ta operacja umożliwia sterowanie dostarczonym uchwytem połączenia z systemem. Jedyne wywołania MQI, które mogą być wykonywane przez wątek inny niż wątek konsumenta, to:

- MQCTL z operacją CTLSP
- MQCTL z operacją CTLSU
- MQDISC-wykonuje operację MQCTL z operacją CTLSP przed rozłączeniem połączenia HConn.

Wartość RC2500 jest zwracana, jeśli wywołanie funkcji API języka IBM MQ zostało wysłane podczas uruchamiania uchwytu połączenia, a wywołanie nie pochodzi z funkcji konsumenta komunikatów.

Jeśli połączenie nie powiedzie się, konwersacja zostanie zatrzymana tak szybko, jak to możliwe. Dlatego możliwe jest, że wywołanie funkcji API IBM MQ w głównym wątku będzie przez pewien czas odbierać kod powrotu RC2500 , po którym następuje kod powrotu RC2009 , gdy połączenie powraca do stanu zatrzymania.

Można to wykonać w funkcji konsumenta. Dla tego samego połączenia, co procedura zwrotna, jej jedynym przeznaczeniem jest anulowanie poprzednio wydanej operacji CTLSP.

Ta opcja nie jest obsługiwana, jeśli aplikacja jest powiązana z niewielowątkową biblioteką IBM MQ .

CTLSW
Rozpocznij odbieranie komunikatów dla wszystkich zdefiniowanych funkcji konsumenta komunikatów dla określonego uchwytu połączenia.

Konsumenty komunikatów działają w tym samym wątku i sterowanie nie jest zwracane do programu wywołującego MQCTL, dopóki:

- zwolniona za pomocą operacji MQCTL CTLSP lub CTLSU, lub
- Wszystkie procedury konsumenta zostały wyrejestrowane lub zawieszone.

Jeśli wszyscy konsumenci są wyrejestrowani lub zawieszeni, wykonywana jest niejawna operacja CTLSP.

Tej opcji nie można używać w procedurze wywołania zwrotnego ani dla bieżącego uchwytu połączenia, ani dla żadnego innego uchwytu połączenia. Jeśli zostanie podjęta próba wywołania, zostanie zwrócony kod powrotu RC2012.

Jeśli w dowolnym momencie podczas operacji CTLSW nie ma zarejestrowanych, niezawieszonych konsumentów, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2446.

Jeśli podczas operacji CTLSW połączenie jest zawieszone, wywołanie MQCTL zwraca kod przyczyny ostrzeżenia RC2521; połączenie pozostaje uruchomione.

Aplikacja może wydać komendę CTLSP lub CTLRE. W tym przypadku operacja CTLRE jest blokami.

Ta opcja nie jest obsługiwana w kliencie jednowątkowym.

CTLSP,
Zatrzymaj odbieranie komunikatów i poczekaj, aż wszyscy konsumenci zakończą swoje operacje, zanim ta opcja zostanie zakończona. Ta operacja zwalnia uchwyt połączenia.

Jeśli ta opcja zostanie wywołana z poziomu procedury zwrotnej, nie będzie obowiązywać do momentu zakończenia procedury. Po zakończeniu procedur konsumenta dla komunikatów, które zostały już odczytane, i po wykonaniu wywołań zatrzymania (na żądanie) procedur zwrotnych, nie są już wywoływane żadne procedury konsumenta komunikatów.

W przypadku wywołania poza procedurą zwrotną sterowanie nie jest zwracane do programu wywołującego, dopóki procedury konsumenta dla komunikatów, które zostały już odczytane, nie zostaną zakończone i po wykonaniu wywołań zwrotnych zatrzymania (na żądanie). Same procedury zwrotne pozostają jednak zarejestrowane.

Ta funkcja nie ma wpływu na komunikaty odczytu z wyprzedzeniem. Należy upewnić się, że konsumenci uruchamiają komendę MQCLOSE (COQSC) z poziomu funkcji zwrotnej, aby określić, czy są dostępne dalsze komunikaty do dostarczenia.

CTLSU
Wstrzymaj odbieranie komunikatów. Ta operacja zwalnia uchwyt połączenia.

Nie ma to wpływu na odczyt komunikatów z wyprzedzeniem dla aplikacji. Jeśli użytkownik zamierza zatrzymać odbieranie komunikatów na długi czas, należy rozważyć zamknięcie kolejki i ponowne jej otwarcie, gdy przetwarzanie musi być kontynuowane.

Jeśli wywołanie zostanie wykonane z procedury zwrotnej, nie będzie ono obowiązywać do momentu wyjścia z tej procedury. Po zakończeniu bieżącej procedury nie będą wywoływane żadne procedury konsumenta komunikatów.

W przypadku wywołania poza wywołaniem zwrotnym sterowanie nie jest zwracane do programu wywołującego, dopóki bieżąca procedura konsumenta nie zostanie zakończona i nie zostanie wywołana żadna procedura.

STLRE
Wznów odbieranie komunikatów.

Ta opcja jest zwykle generowana z głównego wątku aplikacji, ale może być również używana w procedurze zwrotnej w celu anulowania wcześniejszego żądania zawieszenia wydanego w tej samej procedurze.

Jeśli komenda CTLRE jest używana do wznowienia CTLSW, operacja jest blokiem.

PCTLOP (MQCTLO)-wejście

Opcje sterujące działaniem komendy MQCTL

Szczegółowe informacje na temat struktury zawiera sekcja MQCTLO .

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Poniższe kody przyczyny są tymi, które menedżer kolejek może zwrócić dla parametru **Reason** .

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2133
(2133, X'855 ') Nie można załadować modułów usług konwersji danych.

RC2204
(2204, X'89C') Adapter nie jest dostępny.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2374
(2374, X' 946 ') Wyjście API nie powiodło się.

RC2183
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2005
(2005, X'7D5') Niepoprawny parametr długości buforu.

RC2487
(2487, X'9B7') Nie można wywołać procedury zwrotnej

RC2448
(2448, X' 990 ') Nie można wyrejestrować, zawiesić lub wznowić, ponieważ nie ma zarejestrowanego wywołania zwrotnego

RC2486
(2486, X'9B6') W wywołaniu CBREG podano zarówno CallbackFunction , jak i CallbackName , albo jedną z wartości CallbackFunction lub CallbackName , ale nie jest ona zgodna z obecnie zarejestrowaną funkcją zwrotną.

RC2483
(2483, X'9B3') Niepoprawne pole typu CallBack.

RC2219
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2444
(2444, X'98C') Blok opcji jest niepoprawny.

RC2484
(2484, X'9B4') Niepoprawne pole opcji MQCBD.

RC2140
(2140, X'85C') Żądanie oczekiwania zostało odrzucone przez CICS.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2217
(2217, X'8A9') Brak uprawnień do połączenia.

RC2202
(2202, X'89A') wygaszanie połączenia.

RC2203
(2203, X'89B') Połączenie jest zamykane.

RC2207
(2207, X'89F') Błąd identyfikatora korelacji.

RC2016
(2016, X'7E0') Liczba pobrań zablokowana dla kolejki.

RC2351
(2351, X'92F') Globalny konflikt jednostek pracy.

RC2186
(2186, X'88A') Niepoprawna struktura opcji Get-message.

RC2353
(2353, X' 931 ') Uchwyt używany do globalnej jednostki pracy.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2259
(2259, X'8D3') Niespójna specyfikacja przeglądania.

RC2245
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2246
(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

RC2352
(2352, X' 930 ') Globalna jednostka pracy powoduje konflikt z lokalną jednostką pracy.

RC2247
(2247, X'8C7') Opcje zgodności są niepoprawne.

RC2485
(2485, X'9B5') Niepoprawne pole długości MaxMsg

RC2026
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

RC2497
(2497, X'9C1') Określony punkt wejścia funkcji nie został znaleziony w module.

RC2496
(2496, X'9C0') Moduł został znaleziony, ale jest niewłaściwego typu (32-lub 64-bitowy) lub nie jest poprawnym DLL.

RC2495
(2495, X'9BF') Moduł nie został znaleziony w ścieżce wyszukiwania lub nie ma uprawnień do ładowania.

RC2206
(2206, X'89E') Błąd identyfikatora komunikatu.

RC2250
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

RC2331
(2331, X'91B') Użycie znacznika komunikatu jest niepoprawne.

RC2036
(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

RC2037
(2037, X'7F5') Kolejka nie jest otwarta do wprowadzania.

RC2041
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2488
(2488, X'9B8') Niepoprawny kod operacji w wywołaniu API

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2193
(2193, X'891 ') Błąd podczas uzyskiwania dostępu do zestawu danych zestawu stron.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2394
(2394, X'95A') Kolejka ma niepoprawny typ indeksu.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161
(2161, X'871 ') wygaszanie menedżera kolejek.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2069
(2069, X'815 ') Sygnał zaległości dla tego uchwytu.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2109
(2109, X'83D') Wywołanie zablokowane przez program obsługi wyjścia.

RC2072
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2354
(2354, X' 932 ') Niepowodzenie rejestracji w globalnej jednostce pracy.

RC2355
(2355, X' 933 ') Mieszanina wywołań jednostki pracy nie jest obsługiwana.

RC2255
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

RC2090
(2090, X'82A') Niepoprawny odstęp czasu oczekiwania w MQGMO.

RC2256
(2256, X'8D0') Podano niewłaściwą wersję MQGMO.

RC2257
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

RC2298
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQCTL(HCONN : OPERATN : PCTLOP :
                                      CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQCTL            PR                  EXTPROC('MQCTL')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Operation
D OPERATN                       10I 0 VALUE
```

```
     D* Control options
     D PCTLOP                        32A
     D* Completion code
     D CMPCOD                        10I 0
     D* Reason code qualifying CompCode
     D REASON                        10I 0
```

IBM i MQDISC (Rozłączenie menedżera kolejek) w systemie IBM i

Wywołanie MQDISC przerywa połączenie między menedżerem kolejek a aplikacją i jest odwrotnością wywołania MQCONN lub MQCONNX.

Składnia

MQDISC *(HCONN, CMPCOD, REASON)*

Użycie notatek

1. Jeśli wywołanie MQDISC jest wykonywane, gdy aplikacja nadal ma otwarte obiekty, te obiekty są zamykane przez menedżer kolejek z opcjami zamykania ustawionymi na wartość CONONE.
2. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, dyspozycja tych zmian zależy od sposobu zakończenia aplikacji:
   a. Jeśli aplikacja wysyła wywołanie MQDISC przed zakończeniem:
      - W przypadku koordynowanej jednostki pracy menedżera kolejek menedżer kolejek wysyła wywołanie MQCMIT w imieniu aplikacji. Jednostka pracy jest zatwierdzana, jeśli to możliwe, i wycofywana, jeśli nie.
      - W przypadku zewnętrznie koordynowanej jednostki pracy status jednostki pracy nie ulega zmianie. Jednak menedżer kolejek wskazuje, że jednostka pracy powinna zostać zatwierdzona na żądanie koordynatora jednostki pracy.
   b. Jeśli aplikacja zakończy działanie normalnie, ale bez wywołania MQDISC, jednostka pracy zostanie wycofana.
   c. Jeśli aplikacja zakończy działanie *nieprawidłowo* bez wywołania MQDISC, jednostka pracy zostanie wycofana.

Parametry

Wywołanie MQDISC ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście/wyjście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

Po pomyślnym zakończeniu wywołania menedżer kolejek ustawia parametr *HCONN* na wartość, która nie jest poprawnym uchwytem dla środowiska. Wartość ta jest następująca:

HUNUH
Bezużyteczny uchwyt połączenia.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                   CALLP     MQDISC(HCONN : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQDISC           PR                  EXTPROC('MQDISC')
D* Connection handle
D HCONN                         10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQDLTMH (Usunięcie uchwytu komunikatu-Delete message handle) w systemie IBM i

Wywołanie MQDLTMH usuwa uchwyt komunikatu i jest odwrotnością wywołania MQCRTMH.

Składnia

((Hconn, Hmsg, DltMsgHOpts, CompCode, Reason) MQDLTMH

Użycie notatek

1. Tego wywołania można użyć tylko wtedy, gdy sam menedżer kolejek koordynuje jednostkę pracy. Może to być:
   - Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
   - Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby IBM MQ .

   Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja "MQBEGIN (rozpoczęcie jednostki pracy) w systemie IBM i" na stronie 1293.
2. W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiedniego wywołania wycofania zamiast wywołania MQBACK. Środowisko może również obsługiwać niejawne wycofanie spowodowane nieprawidłowym zakończeniem działania aplikacji.
   - W systemie z/OSnależy użyć następujących wywołań:
     - Programy wsadowe (w tym programy IMS w języku DL/I) mogą używać wywołania MQBACK, jeśli jednostka pracy ma wpływ tylko na zasoby IBM MQ . Jeśli jednak jednostka pracy ma wpływ zarówno na zasoby IBM MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRBACK udostępnianego przez usługę RRS (z/OS Recoverable Resource Service). Wywołanie SRRBACK wycofuje zmiany w zasobach należących do menedżerów zasobów, dla których włączono koordynację RRS.
     - Aplikacje CICS muszą używać komendy `EXEC CICS SYNCPOINT ROLLBACK` , aby wycofać jednostkę pracy. Nie należy używać wywołania MQBACK dla aplikacji CICS .
     - Aplikacje IMS (inne niż programy wsadowe DL/I) muszą używać wywołań języka IMS , takich jak `ROLB` , do tworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji IMS (innych niż programy wsadowe DL/I).
   - W systemie IBM itego wywołania należy używać dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **`CMTSCOPE(*JOB)`** nie została wydana dla zadania.
3. Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji "MQDISC (Rozłączenie menedżera kolejek) w systemie IBM i" na stronie 1330 .
4. Gdy aplikacja umieszcza lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Te informacje są powiązane z uchwytem kolejki i obejmują takie elementy, jak:
   - Wartości pól *GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags* w strukturze MQMD.
   - Określa, czy komunikat jest częścią jednostki pracy.

- Dla wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Menedżer kolejek przechowuje trzy zestawy informacji o grupach i segmentach, po jednym zestawie dla każdego z następujących elementów:

- Ostatnie pomyślne wywołanie MQPUT (może być częścią jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
- Ostatnie pomyślne wywołanie MQGET, które przeglądnięło komunikat w kolejce (nie może być częścią jednostki pracy).

Jeśli aplikacja umieszcza lub pobiera komunikaty jako część jednostki pracy, a następnie wycofuje jednostkę pracy, informacje o grupie i segmencie są przywracane do wartości, która była poprzednio:

- Informacje powiązane z wywołaniem MQPUT zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQPUT dla tego uchwytu kolejki w bieżącej jednostce pracy.
- Informacje powiązane z wywołaniem MQGET zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zasięgiem jednostki pracy, nie mają odtworzonych informacji o grupach i segmentach, jeśli jednostka pracy została wycofana.

Przywrócenie poprzedniej wartości informacji o grupie i segmencie po wycofaniu jednostki pracy umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów na kilka jednostek pracy oraz zrestartowanie w odpowiednim punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się. Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną pamięć kolejki. Jednak aplikacja musi zachować wystarczającą ilość informacji, aby można było restartować umieszczanie lub pobieranie komunikatów w odpowiednim miejscu w przypadku wystąpienia awarii systemu.

Szczegółowe informacje na temat restartowania w poprawnym punkcie po awarii systemu zawiera opis opcji PMLOGO w sekcji PMOPT (10-cyfrowa liczba całkowita ze znakiem)oraz opis opcji GMLOGO w sekcji GMOPT (10-cyfrowa liczba całkowita ze znakiem).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

5. Jednostka pracy ma taki sam zasięg jak uchwyt połączenia. Wszystkie wywołania IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wykonane przy użyciu innego uchwytu połączenia (na przykład wywołania wykonane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączenia zawiera sekcja HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe .
6. To wywołanie ma wpływ tylko na komunikaty, które zostały umieszczone lub pobrane jako część bieżącej jednostki pracy.
7. Aplikacja długotrwała, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w jednostce pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może zapełniać kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć się przed taką możliwością, administrator musi ustawić atrybut **MaxUncommittedMsgs** menedżera kolejek na wartość, która jest na tyle niska, aby zapobiec zapełnianiu kolejek przez aplikacje niedziałające, ale na tyle wysoka, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQDLTMH ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **HMSG** .

Jeśli uchwyt komunikatu został utworzony za pomocą HCUNAS, w wątku usuwającym uchwyt komunikatu musi zostać nawiązane poprawne połączenie. W przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2009 .

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście/wyjście

Jest to uchwyt komunikatu, który ma zostać usunięty. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

Po pomyślnym zakończeniu wywołania uchwyt jest ustawiany na niepoprawną wartość dla środowiska. Wartość ta jest następująca:

HMUNUH
Uchwyt komunikatu nie do użycia.

Nie można usunąć uchwytu komunikatu, jeśli inne wywołanie IBM MQ , które przeszło ten sam uchwyt komunikatu, jest w toku.

DLTOPT (MQDMHO)-wejście

Szczegółowe informacje na ten temat zawiera sekcja MQDMHO .

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2462
(2462, X'099E') Niepoprawna struktura opcji usuwania uchwytu komunikatu.

RC2460
(2460, X'099C') Wskaźnik uchwytu komunikatu jest niepoprawny.

RC2499
(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat zawiera sekcja “Kody powrotu dla IBM i (ILE RPG)” na stronie 1471.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDLTMH(HCONN : HMSG : DLTOPT :
                                   CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQDLTMH          PR                  EXTPROC('MQDLTMH')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          20I 0
D* Options that control the action of MQDLTMH
D DLTOPT                        12A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

MQDLTMP-właściwość usuwania komunikatu

Wywołanie MQDLTMP usuwa właściwość z uchwytu komunikatu i jest odwrotnością wywołania MQSETMP.

Składnia

Komenda MQDLTMP *(Hconn, Hmsg, DltPropOpts, Name, CompCode, Reason)*

Parametry

Wywołanie MQDLTMP ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **HMSG** .

Jeśli uchwyt komunikatu został utworzony za pomocą HCUNAS, należy nawiązać poprawne połączenie w wątku usuwającym uchwyt komunikatu. W przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Jest to uchwyt komunikatu zawierający właściwość, która ma zostać usunięta. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

DLTOPT (MQDMPO)-wejście

Szczegółowe informacje można znaleźć w opisie typu danych MQDMPO .

PRNAME (MQCHARV)-wejście

Nazwa właściwości do usunięcia. Więcej informacji na temat nazw właściwości zawiera sekcja Nazwy właściwości .

Znaki wieloznaczne nie są dozwolone w nazwie właściwości.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2471
(2471, X'09A7') Właściwość niedostępna.

RC2421
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'086D') Główne identyfikatory ASID są różne.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2481
(2481, X'09B1') Niepoprawna struktura opcji usuwania właściwości komunikatu.

RC2460
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

RC2499
(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

RC2442
(2442, X'098A') Niepoprawna nazwa właściwości.

RC2111
(2111, X'083F') Niepoprawny identyfikator kodowanego zestawu znaków nazwy właściwości.

RC2195
(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat tych kodów zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQDLTMP(HCONN : HMSG : DLTOPT :
                                    PRNAME : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQDLTMP          PR                  EXTPROC('MQDLTMP')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          20I 0 VALUE
D* Options that control the action of MQDLTMP
D DLTOPT                        12A
D* Property name
D PRNAME                        32A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQGET (pobranie komunikatu) w systemie IBM i

Wywołanie MQGET pobiera komunikat z kolejki lokalnej, która została otwarta za pomocą wywołania MQOPEN.

Składnia

MQGET *(HCONN, HOBJ, MSGDSC, GMO, BUFLEN, BUFFER, DATLEN, CMPCOD, REASON)*

Użycie notatek

1. Pobrany komunikat jest zwykle usuwany z kolejki. To usunięcie może nastąpić w ramach samego wywołania MQGET lub w ramach punktu synchronizacji. Usunięcie komunikatu nie jest wykonywane, jeśli w parametrze **GMO** podano opcję GMBRWF lub GMBRWN (patrz opis pola *GMOPT* w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106).
2. Jeśli opcja GMLK jest określona z jedną z opcji przeglądania, przeglądany komunikat jest blokowany, aby był widoczny tylko dla tego uchwytu.

   Jeśli podano opcję GMUNLK, komunikat poprzednio zablokowany jest odblokowany. W tym przypadku nie jest pobierany żaden komunikat, a parametry **MSGDSC**, **BUFLEN**, **BUFFER** i **DATLEN** nie są sprawdzane ani zmieniane.
3. Jeśli aplikacja wysyłająca wywołanie MQGET działa jako aplikacja IBM MQ MQI client, możliwe jest, że pobrany komunikat zostanie utracony, jeśli podczas przetwarzania wywołania MQGET program IBM MQ MQI client zakończy działanie nieprawidłowo lub połączenie klienta zostanie przerwane. Jest to spowodowane tym, że odpowiednik, który działa na platformie menedżera kolejek i wywołuje wywołanie MQGET w imieniu klienta, nie może wykryć utraty klienta, dopóki odpowiednik nie zwróci komunikatu do klienta. Jest to możliwe po usunięciu komunikatu z kolejki. Taka sytuacja może wystąpić zarówno w przypadku komunikatów trwałych, jak i nietrwałych.

Ryzyko utraty komunikatów w ten sposób można wyeliminować, zawsze pobierając komunikaty w obrębie jednostek pracy (tzn. podając opcję GMSYP w wywołaniu MQGET i używając wywołań MQCMIT lub MQBACK do zatwierdzenia lub wycofania jednostki pracy po zakończeniu przetwarzania komunikatu). Jeśli określono GMSYP, a klient zakończy działanie nieprawidłowo lub połączenie zostanie przerwane, odpowiednik wycofa jednostkę pracy w menedżerze kolejek i komunikat zostanie przywrócony do kolejki.

W zasadzie taka sama sytuacja może wystąpić w przypadku aplikacji działających na platformie menedżera kolejek, ale w tym przypadku okno, w którym komunikat może zostać utracony, jest małe. Jednak podobnie jak w przypadku IBM MQ MQI clients ryzyko można wyeliminować, pobierając komunikat w ramach jednostki pracy.

4. Jeśli aplikacja umieszcza sekwencję komunikatów w określonymw obrębie pojedynczej jednostki pracy, a następnie pomyślnie zatwierdza tę jednostkę pracy, komunikaty stają się dostępne do pobrania w następujący sposób:
   - Jeśli kolejka jest *kolejką niewspółużytkowaną* (czyli kolejką lokalną), wszystkie komunikaty w jednostce pracy stają się dostępne w tym samym czasie.
   - Jeśli kolejka jest *kolejką współużytkowaną*, komunikaty w jednostce pracy stają się dostępne w kolejności, w jakiej zostały umieszczone, ale nie wszystkie w tym samym czasie. Jeśli system jest mocno obciążony, możliwe jest pomyślne pobranie pierwszego komunikatu w jednostce pracy, ale wywołanie MQGET dla drugiego lub kolejnego komunikatu w jednostce pracy może zakończyć się niepowodzeniem z kodem powrotu RC2033. W takim przypadku aplikacja musi odczekać krótki czas, a następnie ponowić operację.
5. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez użycia grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są pewne warunki. Szczegółowe informacje można znaleźć w uwagach dotyczących składni w opisie wywołania MQPUT. Jeśli warunki są spełnione, komunikaty są przedstawiane aplikacji odbierającej w kolejności, w jakiej zostały wysłane, jeśli:
   - Tylko jeden odbiornik otrzymuje komunikaty z kolejki.

     Jeśli istnieją co najmniej dwie aplikacje pobierające komunikaty z kolejki, muszą one uzgodnić z nadawcą mechanizm, który ma być używany do identyfikowania komunikatów należących do sekwencji. Na przykład nadawca może ustawić wszystkie pola `MDCID` w komunikatach w sekwencji na wartość unikalną dla tej sekwencji komunikatów.
   - Odbiorca nie zmienia celowo kolejności pobierania, na przykład podając konkretną wartość `MDMID` lub `MDCID`.

   Jeśli aplikacja wysyłająca umieściła komunikaty jako grupę komunikatów, komunikaty są prezentowane aplikacji odbierającej w poprawnej kolejności, jeśli aplikacja odbierająca określa opcję GMLOGO w wywołaniu MQGET. Więcej informacji na temat grup komunikatów zawiera sekcja:
   - Pole `MDMFL` w strukturze MQMD
   - Opcja PMLOGO w MQPMO
   - Opcja GMLOGO w MQGMO
6. Aplikacje testujące kod sprzężenia zwrotnego FBQUIT w polu `MDFB` parametru **MSGDSC** . Jeśli ta wartość zostanie znaleziona, aplikacja zostanie zakończona. Więcej informacji na ten temat zawiera opis pola `MDFB` w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141 .
7. Jeśli kolejka identyfikowana przez `HOBJ` została otwarta z opcją OOSAVA, a kod zakończenia z wywołania MQGET to CCOK lub CCWARN, kontekst powiązany z uchwytem kolejki `HOBJ` jest ustawiany na kontekst pobranego komunikatu (chyba że ustawiono opcję GMBRWF lub GMBRWN, w którym to przypadku kontekst jest oznaczony jako niedostępny). Tego kontekstu można użyć w kolejnym wywołaniu MQPUT lub MQPUT1 , określając opcje PMPASI lub PMPASA. Umożliwia to przesyłanie kontekstu odebranego komunikatu w całości lub w części do innego komunikatu (na przykład, gdy komunikat jest przekazywany do innej kolejki). Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

8. Jeśli parametr **GMO** zawiera opcję GMCONV, dane komunikatu aplikacji są przekształcane w reprezentację żądaną przez aplikację odbierającą przed umieszczeniem danych w parametrze **BUFFER** :

   - Pole MDFMT w informacjach sterujących komunikatu identyfikuje strukturę danych aplikacji, a pola MDCSI i MDENC w informacjach sterujących komunikatu określają jego identyfikator zestawu znaków i kodowanie.
   - Aplikacja wywołująca wywołanie MQGET określa w polach MDCSI i MDENC parametru **MSGDSC** identyfikator zestawu znaków i kodowanie, na które muszą zostać przekształcone dane komunikatu aplikacji.

   Jeśli konwersja danych komunikatu jest konieczna, konwersja jest wykonywana przez sam menedżer kolejek lub przez wyjście zapisane przez użytkownika, w zależności od wartości pola MDFMT w informacjach sterujących komunikatu:

   - Następujące formaty są przekształcane automatycznie przez menedżer kolejek; są one nazywane formatami wbudowanymi:

FMADMN	FMMDE,
FMCICS	FMPCF,
FMCMD1	FMRMH
FMCMD2	FMRFH
FMDLH	FMRFH2
FMDH	FMSTR,
FMEVNT,	FMTM,
FMIMS	FMXQH
FMIMVS,	

   - Nazwa formatu FMNONE jest wartością specjalną, która wskazuje, że rodzaj danych w komunikacie jest niezdefiniowany. W związku z tym menedżer kolejek nie podejmuje próby konwersji podczas pobierania komunikatu z kolejki.

     Uwaga: Jeśli w wywołaniu MQGET określono wartość GMCONV dla komunikatu o nazwie formatu FMNONE, a zestaw znaków lub kodowanie komunikatu różnią się od podanego w parametrze **MSGDSC** , komunikat jest nadal zwracany w parametrze **BUFFER** (przy założeniu braku innych błędów), ale wywołanie kończy się z kodem zakończenia CCWARN i kodem przyczyny RC2110.

     Wartość FMNONE może być używana, gdy rodzaj danych komunikatu oznacza, że nie wymaga on konwersji, lub gdy aplikacje wysyłające i odbierające uzgodniły między sobą formę, w jakiej powinny być wysyłane dane komunikatu.

   - Wszystkie inne nazwy formatów powodują, że komunikat jest przekazywany do zapisanej przez użytkownika procedury zewnętrznej do konwersji. Wyjście ma taką samą nazwę jak format, z wyjątkiem dodatków specyficznych dla środowiska. Nazwy formatów określone przez użytkownika nie mogą zaczynać się od liter "MQ", ponieważ takie nazwy mogą powodować konflikt z nazwami formatów obsługiwanymi w przyszłości.

   Dane użytkownika w komunikacie mogą być konwertowane między dowolnymi obsługiwanymi zestawami znaków i kodowaniami. Należy jednak pamiętać, że jeśli komunikat zawiera jedną lub więcej struktur nagłówka IBM MQ , nie można dokonać konwersji komunikatu z lub na zestaw znaków, który zawiera znaki dwubajtowe lub wielobajtowe dla żadnego ze znaków poprawnych w nazwach kolejek. Kod przyczyny RC2111 lub RC2115 oznacza, że próba wykonania tej operacji nie powiodła się, a komunikat został zwrócony. Przykładem takiego zestawu znaków jest zestaw znaków Unicode UTF-16 .

   W przypadku powrotu z operacji MQGET następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:

   - BRAK RCNONE

Następujący kod przyczyny wskazuje, że komunikat mógł zostać pomyślnie przekształcony. Aby uzyskać więcej informacji, aplikacja musi sprawdzić pola MDCSI i MDENC w parametrze **MSGDSC** :

- RC2079

Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Uwaga: Interpretacja kodu przyczyny opisana w tym przykładzie jest prawdziwa w przypadku konwersji wykonywanych przez wyjścia zapisane przez użytkownika tylko wtedy, gdy wyjście jest zgodne z wytycznymi przetwarzania.

9. W przypadku wymienionych wcześniej wbudowanych formatów menedżer kolejek może wykonać domyślną konwersję łańcuchów znaków w komunikacie, jeśli określono opcję GMCONV. Konwersja domyślna umożliwia menedżerowi kolejek użycie określonego przez instalację domyślnego zestawu znaków, który przybliża rzeczywisty zestaw znaków podczas konwersji danych łańcuchowych. W rezultacie wywołanie MQGET może zakończyć się pomyślnie z kodem zakończenia CCOK, a nie z kodem CCWARN i kodem przyczyny RC2111 lub RC2115.

   Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji danych łańcuchowych jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Można tego uniknąć, używając w łańcuchu tylko tych znaków, które są wspólne zarówno dla rzeczywistego zestawu znaków, jak i dla domyślnego zestawu znaków.

   Konwersja domyślna dotyczy zarówno danych komunikatu aplikacji, jak i pól znakowych w strukturach MQMD i MQMDE:

   - Domyślna konwersja danych komunikatu aplikacji występuje tylko wtedy, gdy wszystkie poniższe instrukcje są prawdziwe:
     - Aplikacja określa GMCONV.
     - Komunikat zawiera dane, które muszą zostać przekształcone z zestawu znaków lub na zestaw znaków, który nie jest obsługiwany.
     - Konwersja domyślna została włączona podczas instalowania lub restartowania menedżera kolejek.
   - Domyślna konwersja pól znakowych w strukturach MQMD i MQMDE jest wykonywana w razie potrzeby, jeśli dla menedżera kolejek włączono konwersję domyślną. Konwersja jest wykonywana nawet wtedy, gdy opcja GMCONV nie jest określona przez aplikację w wywołaniu MQGET.

10. Parametr **BUFFER** przedstawiony w przykładzie programowania w języku RPG jest zadeklarowany jako łańcuch, co ogranicza maksymalną długość parametru do 256 bajtów. Jeśli wymagany jest większy bufor, parametr musi być zadeklarowany jako struktura lub jako pole w zbiorze fizycznym.

    Zadeklarowanie parametru jako struktury zwiększa maksymalną możliwą długość do 9999 bajtów, natomiast zadeklarowanie parametru jako pola w zbiorze fizycznym zwiększa maksymalną możliwą długość do około 32 kB.

Parametry

Wywołanie MQGET ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość HCONN została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje kolejkę, z której ma zostać pobrany komunikat. Wartość HOBJ została zwrócona przez poprzednie wywołanie MQOPEN. Kolejka musi być otwarta za pomocą co najmniej jednej z następujących opcji (szczegółowe informacje można znaleźć w sekcji “MQOPEN (Open object) w systemie IBM i” na stronie 1362):

- OOOINPS
- OOOINPX
- OOINPQ,
- OOBRW

MSGDSC (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Ta struktura opisuje atrybuty wymaganego komunikatu oraz atrybuty pobranego komunikatu. Szczegółowe informacje można znaleźć w sekcji "MQMD (deskryptor komunikatu) w systemie IBM i" na stronie 1141.

Jeśli wartość parametru BUFLEN jest mniejsza niż długość komunikatu, parametr MSGDSC jest nadal wprowadzany przez menedżer kolejek, niezależnie od tego, czy w parametrze **GMO** określono wartość GMATM (patrz opis pola GMOPT w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106).

Jeśli aplikacja udostępnia deskryptor MQMD version-1 , zwracany komunikat ma przedrostek MQMDE do danych komunikatu aplikacji, ale tylko wtedy, gdy co najmniej jedno z pól w deskryptorze MQMDE ma wartość inną niż domyślna. Jeśli wszystkie pola w MQMDE mają wartości domyślne, MQMDE jest pomijane. Nazwa formatu FMMDE w polu MDFMT w MQMD wskazuje, że istnieje MQMDE.

GMO (MQGMO)-wejście/wyjście

Opcje sterujące działaniem komendy MQGET.

Szczegółowe informacje można znaleźć w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość w bajtach obszaru BUFFER .

Dla komunikatów, które nie mają danych, można podać wartość zero lub jeśli komunikat ma zostać usunięty z kolejki i dane usunięte (w tym przypadku należy podać wartość GMATM).

Uwaga: Długość najdłuższego komunikatu, jaki można odczytać z kolejki, jest określona przez atrybut kolejki **MaxMsgLength** ; patrz "Atrybuty kolejek" na stronie 1410.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wyjście

Obszar, który ma zawierać dane komunikatu.

Bufor musi być wyrównany do granicy odpowiedniej dla rodzaju danych w komunikacie. Wyrównanie 4-bajtowe musi być odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówków IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli wartość BUFLEN jest mniejsza niż długość komunikatu, do pliku BUFFER zostanie przeniesiona możliwie jak najwięcej komunikatów. Dzieje się tak niezależnie od tego, czy w parametrze **GMO** określono GMATM (więcej informacji zawiera opis pola GMOPT w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106).

Zestaw znaków i kodowanie danych w pliku **BUFFER** są podawane w polach MDCSI i MDENC zwracanych w parametrze **MSGDSC** . Jeśli te wartości różnią się od wartości wymaganych przez odbiornik, odbiorca musi dokonać konwersji danych komunikatu aplikacji na zestaw znaków i wymagane kodowanie. Opcja GMCONV może być użyta z zapisanym przez użytkownika wyjściem do wykonania konwersji danych komunikatu (szczegółowe informacje na temat tej opcji zawiera sekcja "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106).

Uwaga: Wszystkie inne parametry w wywołaniu MQGET mają zestaw znaków i kodowanie dla lokalnego menedżera kolejek (podane przez atrybut menedżera kolejek **CodedCharSetId** i ENNAT).

Jeśli wywołanie nie powiedzie się, zawartość buforu może nadal ulec zmianie.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Długość komunikatu.

Jest to długość (w bajtach) danych aplikacji w komunikacie. Jeśli ta długość komunikatu jest większa niż BUFLEN, w parametrze **BUFFER** (to znaczy, że komunikat jest obcięty) zostanie zwrócona tylko BUFLEN bajtów. Jeśli wartość wynosi zero, oznacza to, że komunikat nie zawiera danych aplikacji.

Jeśli wartość parametru BUFLEN jest mniejsza niż długość komunikatu, parametr DATLEN jest nadal wprowadzany przez menedżer kolejek, niezależnie od tego, czy w parametrze **GMO** określono wartość GMATM (więcej informacji zawiera opis pola GMOPT w sekcji "MQGMO (opcje pobierania komunikatów) w systemie IBM i" na stronie 1106). Umożliwia to aplikacji określenie wielkości buforu wymaganego do obsługi danych komunikatu, a następnie ponowne wywołanie wywołania z buforem o odpowiedniej wielkości.

Jeśli jednak określono opcję GMCONV i przekształcone dane komunikatu są zbyt długie, aby zmieścić się w pliku BUFFER, wartość zwracana dla parametru DATLEN jest następująca:

- Długość nieprzekształconych danych dla formatów zdefiniowanych przez menedżera kolejek.

  W takim przypadku, jeśli rodzaj danych powoduje rozszerzenie podczas konwersji, aplikacja musi przydzielić bufor większy niż wartość zwrócona przez menedżer kolejek dla programu DATLEN.
- Wartość zwracana przez wyjście konwersji danych dla formatów zdefiniowanych przez aplikację.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający CMPCOD.

Poniższe kody przyczyny są tymi, które menedżer kolejek może zwrócić dla parametru **REASON** . Jeśli aplikacja określa opcję GMCONV, a program zewnętrzny napisany przez użytkownika jest wywoływany w celu przekształcenia niektórych lub wszystkich danych komunikatu, to wyjście decyduje o tym, jaka wartość jest zwracana dla parametru **REASON** . W rezultacie możliwe są wartości inne niż wartości opisane w dalszej części tej sekcji.

Jeśli CMPCOD to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli CMPCOD ma wartość CCWARN:

RC2120
(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

RC2190
(2190, X'88E') Przekształcony łańcuch jest zbyt duży dla pola.

RC2150
(2150, X'866 ') Niepoprawny łańcuch DBCS.

RC2110
(2110, X'83E') Niepoprawny format komunikatu.

RC2243
(2243, X'8C3') Segmenty komunikatów mają różne identyfikatory CCSID.

RC2244
(2244, X'8C4') Segmenty komunikatów mają różne kodowania.

RC2209
(2209, X'8A1') Brak zablokowanego komunikatu.

RC2119
(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

RC2272
(2272, X'8E0') Dane komunikatu zostały częściowo przekształcone.

RC2145
(2145, X'861 ') Niepoprawny parametr buforu źródłowego.

RC2111
(2111, X'83F') Niepoprawny identyfikator źródłowego kodowanego zestawu znaków.

RC2113
(2113, X'841 ') Nie rozpoznano kodowania dziesiętnego w komunikacie.

RC2114
(2114, X'842 ') Nierozpoznane kodowanie zmiennopozycyjne w komunikacie.

RC2112
(2112, X'840 ') Nierozpoznane kodowanie źródłowe liczby całkowitej.

RC2143
(2143, X'85F') Niepoprawny parametr długości źródła.

RC2146
(2146, X'862 ') Niepoprawny parametr buforu docelowego.

RC2115
(2115, X'843 ') Niepoprawny identyfikator kodowanego zestawu znaków.

RC2117
(2117, X'845 ') Nierozpoznane kodowanie Packed-decimal określone przez odbiornik.

RC2118
(2118, X'846 ') Nierozpoznane kodowanie zmiennopozycyjne określone przez odbiornik.

RC2116
(2116, X'844 ') Nie rozpoznano kodowania docelowej liczby całkowitej.

RC2079
(2079, X'81F') Zwrócono obcięty komunikat (przetwarzanie zakończone).

RC2080
(2080, X'820 ') Zwrócono obcięty komunikat (przetwarzanie nie zostało zakończone).

Jeśli CMPCOD to CCFAIL:

RC2004
(2004, X'7D4') Niepoprawny parametr buforu.

RC2005
(2005, X'7D5') Niepoprawny parametr długości buforu.

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2010
(2010, X'7DA') Niepoprawny parametr długości danych.

RC2016
(2016, X'7E0') Liczba pobrań zablokowana dla kolejki.

RC2186
(2186, X'88A') Niepoprawna struktura opcji Get-message.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2241
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

RC2242
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

RC2259
(2259, X'8D3') Niespójna specyfikacja przeglądania.

RC2245
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2246
(2246, X'8C6') Komunikat pod kursorem nie jest poprawny do pobrania.

RC2247
(2247, X'8C7') Opcje zgodności są niepoprawne.

RC2026
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

RC2250
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

RC2033
(2033, X'7F1') Brak dostępnego komunikatu.

RC2034
(2034, X'7F2') Kursor przeglądania nie jest ustawiony na komunikacie.

RC2036
(2036, X'7F4') Kolejka nie jest otwarta do przeglądania.

RC2037
(2037, X'7F5') Kolejka nie jest otwarta do wprowadzania.

RC2041
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2161
(2161, X'871 ') wygaszanie menedżera kolejek.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2024
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

RC2072
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2255
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

RC2090
(2090, X'82A') Niepoprawny odstęp czasu oczekiwania w MQGMO.

RC2256
(2256, X'8D0') Podano niewłaściwą wersję MQGMO.

RC2257
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQGET(HCONN : HOBJ : MSGDSC : GMO :
C                                     BUFLEN : BUFFER : DATLEN :
C                                     CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQGET            PR                  EXTPROC('MQGET')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object handle
D HOBJ                           10I 0 VALUE
D* Message descriptor
D MSGDSC                        364A
D* Options that control the action of MQGET
D GMO                           112A
D* Length in bytes of the Buffer area
D BUFLEN                         10I 0 VALUE
D* Area to contain the message data
D BUFFER                           *   VALUE
D* Length of the message
D DATLEN                         10I 0
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
```

IBM i MQINQ (zapytanie o atrybuty obiektu) w systemie IBM i

Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

Poprawne są następujące typy obiektów:

- Kolejka
- Lista nazw
- Definicja procesu
- Menedżer kolejek

- “Składnia” na stronie 1346
- “Użycie notatek” na stronie 1346
- “Parametry” na stronie 1347

- "Deklaracja RPG" na stronie 1354

Składnia

MQINQ *(HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)*

Użycie notatek

1. Zwracane wartości są obrazem stanu wybranych atrybutów. Nie ma gwarancji, że atrybuty nie zostaną zmienione, zanim aplikacja będzie mogła działać na zwróconych wartościach.
2. Po otwarciu kolejki modelowej tworzona jest dynamiczna kolejka lokalna. Dzieje się tak nawet wtedy, gdy kolejka modelowa jest otwierana w celu uzyskania informacji o jej atrybutach.

   Atrybuty kolejki dynamicznej (z pewnymi wyjątkami) są takie same jak atrybuty kolejki modelowej w momencie tworzenia kolejki dynamicznej. Jeśli następnie w tej kolejce zostanie użyte wywołanie MQINQ, menedżer kolejek zwróci atrybuty kolejki dynamicznej, a nie atrybuty kolejki modelowej. Szczegółowe informacje o tym, które atrybuty kolejki modelowej są dziedziczone przez kolejkę dynamiczną, zawiera Tabela 1 .
3. Jeśli obiekt, którego dotyczy zapytanie, jest kolejką aliasową, wartości atrybutów zwracane przez wywołanie MQINQ są wartościami kolejki aliasowej, a nie wartościami kolejki podstawowej, na którą alias jest tłumaczony.
4. Jeśli obiekt, którego dotyczy zapytanie, jest kolejką klastra, atrybuty, które można sprawdzić, zależą od sposobu otwarcia kolejki:
   - Jeśli kolejka klastra jest otwarta do odpytywania oraz co najmniej jedna z wartości wejściowych, przeglądania lub zestawu, aby otwarcie powiodło się, musi istnieć lokalna instancja kolejki klastra. W tym przypadku atrybuty, do których można uzyskać dostęp, są poprawne dla kolejek lokalnych.
   - Jeśli kolejka klastra jest otwarta tylko na potrzeby wykonywania zapytań lub na potrzeby wykonywania zapytań i danych wyjściowych, mogą być wysyłane zapytania tylko do następujących atrybutów. W tym przypadku atrybut **QType** ma wartość QTCLUS:
     - CAQD
     - CAQN
     - MADBND
     - IADPER,
     - IADPRI
     - IAIPUT (wprowadzanie)
     - IAQTYP

     Jeśli kolejka klastra jest otwarta bez stałego powiązania (czyli z nazwą OOBNDN określoną w wywołaniu MQOPEN lub OOBNDQ określoną, gdy atrybut **DefBind** ma wartość BNDNOT), kolejne wywołania MQINQ dla kolejki mogą pytać o różne instancje kolejki klastra, chociaż zwykle wszystkie instancje mają takie same wartości atrybutów.

   Więcej informacji na temat kolejek klastra zawiera sekcja Konfigurowanie klastra menedżera kolejek.
5. Jeśli pewna liczba atrybutów ma zostać sprawdzona, a następnie niektóre z nich mają zostać ustawione za pomocą wywołania MQSET, wygodnie może być umieścić na początku tablic selektora atrybuty, które mają zostać ustawione, tak aby te same tablice (ze zmniejszoną liczbą wystąpień) mogły być używane dla wywołania MQSET.
6. Jeśli wystąpi więcej niż jedna sytuacja ostrzegawcza (patrz parametr **CMPCOD**), zwracany jest *pierwszy* kod przyczyny z poniższej listy, który ma zastosowanie:
   a. RC2068
   b. RC2022
   c. RC2008
7. Więcej informacji na temat atrybutów obiektu zawiera sekcja:

- “Atrybuty kolejek” na stronie 1410
- “Atrybuty listy nazw” na stronie 1440
- “Atrybuty definicji procesów w systemie IBM i” na stronie 1441
- “Atrybuty menedżera kolejek w systemie IBM i” na stronie 1443

8. Nowa kolejka lokalna SYSTEM.ADMIN.COMMAND.EVENT jest używany do kolejkowania komunikatów generowanych za każdym razem, gdy wydawane są komendy. Komunikaty są umieszczane w tej kolejce dla większości komend, w zależności od sposobu ustawienia atrybutu menedżera kolejek CMDEV:
   - Włączono-komunikaty zdarzeń komend są generowane i umieszczane w kolejce dla wszystkich pomyślnych komend.
   - NODISPLAY-komunikaty zdarzeń komend są generowane i umieszczane w kolejce dla wszystkich pomyślnych komend innych niż DISPLAY (MQSC) i Inquire (PCF).
   - DISABLED-komunikaty zdarzeń komend nie są generowane (jest to początkowa wartość domyślna menedżera kolejek).

Parametry

Wywołanie MQINQ ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje obiekt (dowolnego typu) z wymaganymi atrybutami. Uchwyt musi zostać zwrócony przez poprzednie wywołanie MQOPEN, które określało opcję OOINQ.

SELCNT (10-cyfrowa liczba całkowita ze znakiem)-wejście

Liczba selektorów.

Jest to liczba selektorów, które są dostarczane w tablicy *SELS* . Jest to liczba atrybutów, które mają zostać zwrócone. Zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

SELS (10-cyfrowa liczba całkowita ze znakiem x SELCNT)-dane wejściowe

Tablica selektorów atrybutów.

Jest to tablica selektorów atrybutów **SELCNT** . Każdy selektor identyfikuje atrybut (liczbę całkowitą lub znak) z wymaganą wartością.

Każdy selektor musi być poprawny dla typu obiektu reprezentowanego przez *HOBJ* , w przeciwnym razie wywołanie zakończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2067.

W przypadku kolejek specjalnych:

- Jeśli selektor nie jest poprawny dla kolejek *dowolnego typu* , wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i kodem przyczyny RC2067.
- Jeśli selektor ma zastosowanie tylko do kolejek typu lub typów innych niż kolejki obiektu, wywołanie powiedzie się z kodem zakończenia CCWARN i kodem przyczyny RC2068.
- Jeśli kolejka, której dotyczy zapytanie, jest kolejką klastra, poprawne selektory zależą od sposobu rozstrzygnięcia kolejki. Więcej informacji na ten temat zawiera uwaga dotycząca użycia 4.

Selektory można określić w dowolnej kolejności. Wartości atrybutów odpowiadające selektorom atrybutów całkowitych (IA* selectors) są zwracane w produkcie *INTATR* w tej samej kolejności,

w jakiej selektory te występują w produkcie *SELS*. Wartości atrybutów odpowiadające selektorom atrybutów znakowych (CA* selectors) są zwracane w produkcie *CHRATR* w tej samej kolejności, w jakiej występują te selektory. Selektory IA* mogą być przeplatane z selektorami CA*; ważna jest tylko kolejność względna w obrębie każdego typu.

Uwaga:

1. Selektory atrybutów całkowitych i znakowych są przydzielane w dwóch różnych zakresach; selektory IA* rezydują w zakresie od IAFRST do IALAST, a selektory CA* w zakresie od CAFRST do CALAST.

   Dla każdego zakresu stałe IALSTU i CALSTU definiują najwyższą wartość, którą akceptuje menedżer kolejek.
2. Jeśli najpierw wystąpią wszystkie selektory IA*, można użyć tych samych numerów elementów do adresowania odpowiednich elementów w macierzach *SELS* i *INTATR* .

Atrybuty, do których można utworzyć zapytanie, są wymienione w poniższych tabelach. W przypadku selektorów CA* stała definiująca długość wynikowego łańcucha w bajtach w *CHRATR* jest podana w nawiasach.

Tabela 746. Selektory atrybutów MQINQ dla kolejek

Selektor	Opis	Uwaga
CAALTD	Data ostatniej zmiany (LNDATE).	1
CAALTT	Czas ostatniej zmiany (LNTIME).	1
KABRQN	Nadmierna liczba nazw backout-requeue (LNQN).	5
KABASQ	Nazwa kolejki, na którą alias jest tłumaczony (LNQN).	
CACFSN (pamięć podręczna)	Nazwa struktury narzędzia CF (LNCFSN).	3
Numer pamięci podręcznej	Nazwa klastra (LNCLUN).	1
CACLNL (pamięć podręczna)	Lista nazw klastrów (LNNLN).	1
CACRTD (pamięć podręczna)	Data utworzenia kolejki (LNCRTD).	
CACRTT (pamięć podręczna)	Czas utworzenia kolejki (LNCRTT).	
CAINIK@ item: intab	Nazwa kolejki inicjującej (LNQN).	
KAPRON	Nazwa definicji procesu (LNPRON).	
CAQD	Opis kolejki (LNQD).	
CAQN	Nazwa kolejki (LNQN).	
KARQMN	Nazwa zdalnego menedżera kolejek (LNQMN).	
KARQN	Nazwa kolejki zdalnej znana w zdalnym menedżerze kolejek (LNQN).	
KATRGD	Dane wyzwalacza (LNTRGD).	5

Tabela 746. Selektory atrybutów MQINQ dla kolejek (kontynuacja)

Selektor	Opis	Uwaga
KAXQN	Nazwa kolejki transmisji (LNQN).	
IABTHR	Próg wycofania.	5
IACDEP	Liczba komunikatów w kolejce.	
MADBND	Powiązanie domyślne.	1
IADINP	Domyślna opcja open-for-input.	5
IADPER,	Domyślna trwałość komunikatu.	
IADPRI	Domyślny priorytet komunikatu.	5
IADEFT	Typ definicji kolejki.	
IADIST	Obsługa listy dystrybucyjnej.	2
IAHGB	Określa, czy ma być harden backout count.	5
IAIGET	Określa, czy operacje pobierania są dozwolone.	
IAIPUT (wprowadzanie)	Określa, czy operacje umieszczania (put) są dozwolone.	
IAMLEN	Maksymalna długość komunikatu.	
IAMDEP	Maksymalna liczba komunikatów dozwolonych w kolejce.	
IAMDS	Określa, czy priorytet komunikatu jest istotny.	5
IAOIC,	Liczba wywołań MQOPEN z otwartą kolejką dla wejścia.	
MAOOC,	Liczba wywołań MQOPEN z otwartą kolejką dla danych wyjściowych.	
IAQDHE	Atrybut sterujący dla zdarzeń dużego zapełnienia kolejki.	4, 5
IAQDHL	Górny limit głębokości kolejki.	4, 5
BEZCZYNNY	Atrybut sterujący dla zdarzeń niskiego zapełnienia kolejki.	4, 5
Biblioteka IAQDLL	Dolny limit głębokości kolejki.	4, 5
IAQDME	Atrybut sterujący dla zdarzeń maksymalnej głębokości kolejki.	4, 5
IAQSI	Limit czasu obsługi kolejki.	4, 5
IAQSIE	Atrybut sterujący dla zdarzeń odstępu czasu usługi kolejki.	4, 5
IAQTYP	Typ kolejki.	
IAQSGD	Dyspozycja grupy współużytkowania kolejek.	3
IARLCAŁK	Odstęp czasu przechowywania kolejki.	5
IASCOP	Zasięg definicji kolejki.	4, 5
IASHAR	Określa, czy kolejka może być współużytkowana dla danych wejściowych.	
IATRGC	Element sterujący wyzwalacza.	
IATRGD	Wyzwalacz uruchamiany zapełnieniem.	5
IATRGP	Priorytet komunikatu progowego dla wyzwalaczy.	5

Tabela 746. Selektory atrybutów MQINQ dla kolejek (kontynuacja)		
Selektor	**Opis**	**Uwaga**
IATRGT	Typ wyzwalacza.	
MAUSAG	Użycie.	
CLWLUSEQ	Użyj kolejek zdalnych.	

Uwaga:

1. Obsługiwane na następujących platformach:
   - AIX
   - IBM i
   - Windows
   - z/OS

   i dla IBM MQ MQI clients połączonych z tymi systemami.
2. Obsługiwane na następujących platformach:
   - AIX
   - IBM i
   - Windows

   i dla klientów IBM MQ połączonych z tymi systemami.
3. z/OS Obsługiwane w systemie z/OS.
4. z/OS Nieobsługiwane w systemie z/OS.
5. Nieobsługiwane w systemie VSE/ESA.

Tabela 747. Selektory atrybutów MQINQ dla list nazw		
Selektor	**Opis**	**Uwaga**
CAALTD	Data ostatniej zmiany (LNDATE)	1
CAALTT	Czas ostatniej zmiany (LNTIME)	1
CALSTD	Opis listy nazw (LNNLD)	1
CALSTN	Nazwa obiektu listy nazw (LNNLN)	1
KANKI	Nazwy na liście nazw (LNQN x *Liczba nazw na liście)*	1
IANAMC	Liczba nazw na liście nazw	1
IAQSGD	Dyspozycja grupy współużytkowania kolejki	3

Tabela 748. Selektory atrybutów MQINQ dla definicji procesów		
Selektor	**Opis**	**Uwaga**
CAALTD	Data ostatniej zmiany (LNDATE)	1
CAALTT	Czas ostatniej zmiany (LNTIME)	1
CAPPI	Identyfikator aplikacji (LNPROA)	5

Tabela 748. Selektory atrybutów MQINQ dla definicji procesów (kontynuacja)		
Selektor	**Opis**	**Uwaga**
CENVD	Dane środowiska (LNPROE)	5
KOD	Opis definicji procesu (LNPROD)	5
KAPRON	Nazwa definicji procesu (LNPRON)	5
CUSRD	Dane użytkownika (LNPROU)	5
MAAPPT	Typ aplikacji	5
IAQSGD	Dyspozycja grupy współużytkowania kolejki	3

Tabela 749. Selektory atrybutów MQINQ dla menedżera kolejek		
Selektor	**Opis**	**Uwaga**
CAALTD	Data ostatniej zmiany (LNDATE)	1
CAALTT	Czas ostatniej zmiany (LNTIME)	1
CACADX (pamięć podręczna)	Nazwa wyjścia definicji kanału automatycznego (LNEXN)	1
CACLWD	Dane przekazywane do wyjścia obciążenia klastra (LNEXDA)	1
CACLWX (pamięć podręczna)	Nazwa wyjścia obciążenia klastra (LNEXN)	1
CACMDQ (pamięć podręczna)	Nazwa kolejki wejściowej komendy systemowej (LNQN)	5
KADLQ	Nazwa kolejki niedostarczonych komunikatów (LNQN)	5
KADXQN	Domyślna nazwa kolejki transmisji (LNQN)	5
CAQMD	Opis menedżera kolejek (LNQMD)	5
Identyfikator CAQMID	Identyfikator menedżera kolejek (LNQMID)	1
CAQMN	Nazwa lokalnego menedżera kolejek (LNQMN)	5
CAQSGN	Nazwa grupy współużytkowania kolejki (LNQSGN)	3
KARPN	Nazwa klastra, dla którego menedżer kolejek udostępnia usługi repozytorium (LNQMN)	1
KARPNL	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których menedżer kolejek udostępnia usługi repozytorium (LNNLN)	1
CMDEV	Atrybut sterujący określający, czy komunikaty generowane podczas wydawania komend są umieszczane w kolejce	8
MAAUTE	Atrybut sterujący dla zdarzeń uprawnień	4, 5
IAKAD	Atrybut sterujący dla automatycznej definicji kanału	2
IACADE	Atrybut sterujący dla zdarzeń automatycznej definicji kanału	2
IACLXQ,	Domyślny typ kolejki transmisji klastra	4

Tabela 749. Selektory atrybutów MQINQ dla menedżera kolejek (kontynuacja)		
Selektor	**Opis**	**Uwaga**
IACLWL	Długość obciążenia klastra	1
IACCSI,	Identyfikator kodowanego zestawu znaków	5
IACMDLStencils	Poziom komend obsługiwany przez menedżer kolejek	5
IACFGE	Atrybut sterujący dla zdarzeń konfiguracji	3
IADIST	Obsługa listy dystrybucyjnej	2
MAINHE	Atrybut sterujący dla zdarzeń blokowania	4, 5
IALKLE	Atrybut sterujący dla zdarzeń lokalnych	4, 5
IAMHND	Maksymalna liczba uchwytów	5
IAMLEN	Maksymalna długość komunikatu	5
IAMPRI	Maksymalny priorytet	5
IAMUNC	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy	5
IAPFME	Atrybut sterujący dla zdarzeń wydajności	4, 5
IAPLAT	Platforma, na której rezyduje menedżer kolejek	5
IARMTE	Atrybut sterujący dla zdarzeń zdalnych	4, 5
IASSE	Atrybut sterujący dla zdarzeń uruchomienia i zatrzymania	4, 5
IASYNC	Dostępność punktu synchronizacji	5
IATRLFT,	Czas życia nieużywanych tematów nieadministracyjnych	
IATRGI	Interwał wyzwalacza	5

IACNT (10-cyfrowa liczba całkowita ze znakiem)-wejście

Liczba atrybutów całkowitych.

Jest to liczba elementów w tablicy INTATR . Zero jest poprawną wartością.

Jeśli jest to co najmniej liczba selektorów IA* w parametrze **SELS** , zwracane są wszystkie żądane atrybuty całkowitoliczbowe.

INTATR (10-cyfrowa liczba całkowita ze znakiem x IACNT)-dane wyjściowe

Tablica atrybutów całkowitych.

Jest to tablica wartości atrybutów całkowitoliczbowych *IACNT* .

Wartości atrybutów całkowitych są zwracane w tej samej kolejności, co selektory IA* w parametrze **SELS** . Jeśli tablica zawiera więcej elementów niż liczba selektorów IA*, nadmiarowe elementy pozostają niezmienione.

Jeśli parametr HOBJ reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, dla odpowiedniego elementu w tablicy INTATR zwracana jest konkretna wartość IAVNA.

CALEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość buforu atrybutów znakowych.

Jest to długość parametru **CHRATR** w bajtach.

Musi to być co najmniej suma długości żądanych atrybutów znakowych (patrz sekcja SELS). Zero jest poprawną wartością.

CHRATR (1-bajtowy łańcuch znaków x CALEN)-dane wyjściowe

Atrybuty znakowe.

Jest to bufor, w którym zwracane są atrybuty znakowe, połączone ze sobą. Długość buforu jest określona przez parametr **CALEN** .

Atrybuty znakowe są zwracane w tej samej kolejności, co selektory CA* w parametrze **SELS** . Długość każdego łańcucha atrybutu jest ustalona dla każdego atrybutu (patrz sekcja SELS), a jego wartość jest uzupełniana po prawej stronie odstępami, jeśli jest to konieczne. Jeśli bufor jest większy niż wymagany, aby zawierał wszystkie żądane atrybuty znakowe (w tym dopełnianie), zwrócone bajty wykraczające poza ostatnią wartość atrybutu pozostają niezmienione.

Jeśli HOBJ reprezentuje kolejkę, ale selektor atrybutu nie ma zastosowania do tego typu kolejki, jako wartość tego atrybutu w programie CHRATRzwracany jest łańcuch znaków składający się wyłącznie z gwiazdek (*).

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający CMPCOD.

Jeśli CMPCOD to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2008
(2008, X'7D8') Brak wystarczającej ilości miejsca dla atrybutów znakowych.

RC2022
(2022, X'7E6') Niewystarczająca ilość miejsca dla atrybutów całkowitoliczbowych.

RC2068
(2068, X'814 ') Selektor nie dotyczy typu kolejki.

Jeśli *CMPCOD* to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2006
(2006, X'7D6') Niepoprawna długość atrybutów znakowych.

RC2007
(2007, X'7D7') Niepoprawny łańcuch atrybutów znaku.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2021
(2021, X'7E5') Niepoprawna liczba atrybutów liczby całkowitej.

RC2023
(2023, X'7E7') Niepoprawna tablica atrybutów całkowitych.

RC2038
(2038, X'7F6') Kolejka nie jest otwarta do odpytywania.

RC2041
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2065
(2065, X'811 ') Niepoprawna liczba selektorów.

RC2067
(2067, X'813 ') Niepoprawny selektor atrybutu.

RC2066
(2066, X'812 ') Zbyt duża liczba selektorów.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQINQ(HCONN : HOBJ : SELCNT :
C                                     SELS(1) : IACNT : INTATR(1) :
C                                     CALEN : CHRATR : CMPCOD :
C                                     REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQINQ            PR                  EXTPROC('MQINQ')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Object handle
D HOBJ                          10I 0 VALUE
D* Count of selectors
D SELCNT                        10I 0 VALUE
D* Array of attribute selectors
D SELS                          10I 0
D* Count of integer attributes
D IACNT                         10I 0 VALUE
D* Array of integer attributes
D INTATR                        10I 0
D* Length of character attributes buffer
D CALEN                         10I 0 VALUE
D* Character attributes
```

```
     D CHRATR                         *   VALUE
     D* Completion code
     D CMPCOD                       10I 0
     D* Reason code qualifying CMPCOD
     D REASON                       10I 0
```

IBM i MQINQMP (właściwość komunikatu zapytania) w systemie IBM i

Wywołanie MQINQMP zwraca wartość właściwości komunikatu.

- “Składnia” na stronie 1355
- “Parametry” na stronie 1355
- “Deklaracja RPG” na stronie 1359

Składnia

Komenda MQINQMP *(Hconn, Hmsg, InqPropOpts, Name, PropDesc, Type, ValueLength, Value, DataLength, CompCode, Reason)*

Parametry

Wywołanie MQINQMP ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **Hmsg** .

Jeśli uchwyt komunikatu został utworzony za pomocą HCUNAS, w wątku należy nawiązać poprawne połączenie, sprawdzając właściwość uchwytu komunikatu. W przeciwnym razie wywołanie zakończy się niepowodzeniem z błędem RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Jest to uchwyt komunikatu, którego dotyczy zapytanie. Wartość została zwrócona przez poprzednie wywołanie funkcji **MQCRTMH** .

INQOPT (MQIMPO)-wejście

Szczegółowe informacje można znaleźć w opisie typu danych MQIMPO .

PRNAME (MQCHARV)-wejście

Opisuje nazwę właściwości, do której ma zostać wykonane zapytanie.

Jeśli nie można znaleźć właściwości o tej nazwie, wywołanie kończy się niepowodzeniem z kodem przyczyny RC2471.

Na końcu nazwy właściwości można użyć znaku procentu (%). Znak wieloznaczny zastępuje zero lub więcej znaków, łącznie ze znakiem kropki (.). Dzięki temu aplikacja może uzyskać informacje o wartości wielu właściwości. Wywołaj komendę MQINQMP z opcją IPINQF, aby uzyskać pierwszą zgodną właściwość, a następnie ponownie z opcją IPINQN, aby uzyskać następną zgodną właściwość. Jeśli nie ma więcej zgodnych właściwości, wywołanie kończy się niepowodzeniem z kodem powrotu RC2471. Jeśli pole *ReturnedName* struktury InqPropOpts zostało zainicjowane z adresem lub przesunięciem dla zwróconej nazwy właściwości, jest to wykonywane po zwróceniu przez MQINQMP nazwy właściwości, która została dopasowana. Jeśli pole *VSBufSize* w pliku *ReturnedName* w strukturze InqPropOpts jest mniejsze niż długość zwróconej nazwy właściwości, kod zakończenia jest ustawiany na wartość CCFAIL z przyczyną RC2465.

Właściwości, które mają znane synonimy, są zwracane w następujący sposób:

1. Właściwości z przedrostkiem mqps. są zwracane z nazwą właściwości IBM MQ . Na przykład "MQTopicString" jest zwróconą nazwą, a nie "mqps.Top".

2. Właściwości z przedrostkiem jms. lub "mcd." są zwracane jako nazwa pola nagłówka JMS . Na przykład "JMSExpiration" jest zwróconą nazwą, a nie "jms.Exp".
3. Właściwości z przedrostkiem usr. są zwracane bez tego przedrostka. Na przykład zwracana jest wartość "Color" zamiast "usr.Color".

Właściwości z synonimami są zwracane tylko raz.

W języku programowania RPG zdefiniowane są następujące zmienne makra w celu sprawdzenia wszystkich właściwości i wszystkich właściwości rozpoczynających się od "usr.":

INQALL,
Sprawdź wszystkie właściwości komunikatu.

INQUSR,
Sprawdź wszystkie właściwości komunikatu rozpoczynające się od łańcucha usr. Zwrócona nazwa jest zwracana bez "usr." prefiks.

Jeśli określono parametr IPINQN, ale nazwa uległa zmianie od poprzedniego wywołania lub jest to pierwsze wywołanie, to domyślnie przyjmowany jest parametr IPINQF.

Więcej informacji na temat używania nazw właściwości zawiera sekcja Nazwy właściwości i sekcja Ograniczenia dotyczące nazw właściwości .

PRPDSC (MQPD)-dane wyjściowe

Ta struktura jest używana do definiowania atrybutów właściwości, w tym informacji o tym, co się stanie, jeśli właściwość nie jest obsługiwana, kontekstu komunikatu, do którego należy właściwość oraz komunikatów, do których właściwość powinna zostać skopiowana. Szczegółowe informacje na temat tej struktury zawiera sekcja MQPD .

TYPE (10-cyfrowa liczba całkowita ze znakiem)-wejście/wyjście

Po powrocie z wywołania MQINQMP ten parametr jest ustawiany na typ danych *Wartość*. Typ danych może być dowolny z następujących typów:

TYPBOL
Wartość boolowska.

TYPBST
Łańcuch bajtowy.

TYPI8
8-bitowa liczba całkowita ze znakiem.

TYPI16
16-bitowa liczba całkowita ze znakiem.

TYPI32
32-bitowa liczba całkowita ze znakiem.

TYPI64
64-bitowa liczba całkowita ze znakiem.

TYPF32
32-bitowa liczba zmiennopozycyjna.

TYPF64
64-bitowa liczba zmiennopozycyjna.

TYPSTR
Łańcuch znaków.

TYPNUL
Właściwość istnieje, ale ma wartość null.

Jeśli typ danych wartości właściwości nie zostanie rozpoznany, zostanie zwrócona wartość TYPSTR i w obszarze *Wartość* zostanie umieszczona reprezentacja łańcuchowa wartości. Reprezentację łańcuchową typu danych można znaleźć w polu *IPTYP* parametru *IPOPT* . Zwrócony został kod zakończenia ostrzeżenia z kodem przyczyny RC2467.

Dodatkowo, jeśli podano opcję IPCTYP, żądana jest konwersja wartości właściwości. Użyj opcji *Typ* jako danych wejściowych, aby określić typ danych, który ma być zwracany przez właściwość. Szczegółowe informacje na temat konwersji typu danych można znaleźć w opisie opcji IPCTYP w pliku “MQIMPO (opcje właściwości komunikatu zapytania) w systemie IBM i” na stronie 1134 .

Jeśli konwersja typów nie jest żądana, na wejściu można użyć następującej wartości:

TYPAST
Wartość właściwości jest zwracana bez przekształcania jej typu danych.

VALLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Długość obszaru wartości w bajtach.

Należy podać zero dla właściwości, dla których nie jest wymagana wartość zwracana. Mogą to być właściwości zaprojektowane przez aplikację tak, aby miały wartość NULL lub pusty łańcuch. Należy również podać zero, jeśli podano opcję IPQLEN; w tym przypadku nie jest zwracana żadna wartość.

VALUE (1-bajtowy bit stringxVALLEN)-dane wyjściowe

Jest to obszar, w którym ma być zawarta wartość właściwości, do której wysłano zapytanie. Bufor powinien być wyrównany do granicy odpowiedniej dla zwracanej wartości. Jeśli ta opcja nie zostanie użyta, może to spowodować błąd, gdy wartość zostanie później użyta.

Jeśli wartość *VALLEN* jest mniejsza niż długość wartości właściwości, do *VALUE* jest przenoszona jak najwięcej wartości właściwości, a wywołanie kończy się niepowodzeniem z kodem zakończenia CCFAIL i przyczyną RC2469.

Zestaw znaków danych w polu *VALUE* jest określony przez pole IPRETCSI w parametrze INQOPT. Kodowanie danych w *VALUE* jest nadawane przez pole IPRETENC w parametrze INQOPT.

Jeśli parametr *VALLEN* ma wartość zero, wartość VALUE nie jest przywoływana.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Jest to długość (w bajtach) rzeczywistej wartości właściwości zwróconej w obszarze *Wartość* .

Jeśli wartość *DataLength* jest mniejsza niż długość wartości właściwości, wartość *DataLength* jest nadal wprowadzana po powrocie z wywołania MQINQMP. Dzięki temu aplikacja może określić wielkość buforu wymaganą do dostosowania wartości właściwości, a następnie ponownie wywołać wywołanie z buforem o odpowiedniej wielkości.

Mogą być również zwracane następujące wartości.

Jeśli parametr *Typ* ma wartość TYPSTR lub TYPBST:

WLEMP
Właściwość istnieje, ale nie zawiera znaków ani bajtów.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CompCode*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość CCWARN:

RC2492
(2492, X'09BC') Zwrócona nazwa właściwości nie została przekształcona.

RC2466
(2466, X'09A2') Wartość właściwości nie została przekształcona.

RC2467
(2467, X'09A3') Typ danych właściwości nie jest obsługiwany.

RC2421
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'0852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'086D') Główne identyfikatory ASID są różne.

RC2004
(2004, X'07D4') Niepoprawny parametr wartości.

RC2005
(2005, X'07D5') Niepoprawny parametr długości wartości.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2010
(2010, X'07DA') Niepoprawny parametr długości danych.

RC2464
(2464, X'09A0') Niepoprawna struktura opcji właściwości komunikatu.

RC2460
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

RC2499
(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2064
(2046, X'07F8') Opcje są niepoprawne lub niespójne.

RC2482
(2482, X'09B2') Niepoprawna struktura deskryptora właściwości.

RC2470
(2470, X'09A6') Konwersja rzeczywistego na żądany typ danych nie jest obsługiwana.

RC2442
(2442, X'098A') Niepoprawna nazwa właściwości.

RC2465
(2465, X'09A1') Nazwa właściwości jest zbyt duża dla zwróconego buforu nazw.

RC2471
(2471, X'09A7) Właściwość niedostępna.

RC2469
(2469, X'09A5') Wartość właściwości jest zbyt duża dla obszaru wartości.

RC2472
(2472, X'09A8') Wystąpił błąd formatu liczb w danych wartości.

RC2473
(2473, X'09A9') Niepoprawny żądany typ właściwości.

RC2111
(2111, X'083F') Niepoprawny identyfikator kodowanego zestawu znaków nazwy właściwości.

RC2071
(2071, X'0871 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'0893 ') Wystąpił nieoczekiwany błąd.

Szczegółowe informacje na temat tych kodów znajdują się w następujących sekcjach:

- Komunikaty systemu IBM MQ for z/OS , kody zakończenia i kody przyczyny dla IBM MQ for z/OS
- Komunikaty i kody przyczyny dla wszystkich innych platform IBM MQ

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQINQMP(HCONN : HMSG : INQOPT :
                                        PRNAME : PRPDSC : TYPE :
                                        VALLEN : VALUE : DATLEN :
                                        CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQINQMP          PR                  EXTPROC('MQINQMP')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          20I 0 VALUE
D* Options that control the action of MQINQMP
D INQOPT                        72A
D* Property name
D PRNAME                        32A
D* Property descriptor
D PRPDSC                        24A
D* Property data type
D TYPE                          10I 0
D* Length in bytes of the Value area
D VALLEN                        10I 0 VALUE
D* Property value
D VALUE                           *   VALUE
D* Length of the property value
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQMHBUF (przekształcenie uchwytu komunikatu w bufor) w systemie IBM i

MQMHBUF przekształca uchwyt komunikatu w bufor i jest odwrotnością wywołania MQBUFMH.

- “Składnia” na stronie 1359
- “Użycie notatek” na stronie 1360
- “Parametry” na stronie 1360
- “Deklaracja RPG” na stronie 1362

Składnia

KQMHBUF *(Hconn, Hmsg, MsgHBufOpts, Name, MsgDesc, BufferLength, Buffer, DataLength, CompCode, Reason)*

Użycie notatek

MQMHBUF przekształca uchwyt komunikatu w bufor.

Można go używać z wyjściem MQGET API w celu uzyskania dostępu do pewnych właściwości, korzystając z interfejsów API właściwości komunikatu, a następnie przekazać te właściwości w buforze z powrotem do aplikacji zaprojektowanej do używania nagłówków MQRFH2 zamiast uchwytów komunikatów.

To wywołanie jest odwrotnością wywołania MQBUFMH, którego można użyć do przeanalizowania właściwości komunikatu z buforu do uchwytu komunikatu.

Parametry

Wywołanie MQMHBUF ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość *HCONN* musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **HMSG** .

Jeśli uchwyt komunikatu został utworzony za pomocą HCUNAS, należy nawiązać poprawne połączenie w wątku usuwającym uchwyt komunikatu. Jeśli poprawne połączenie nie zostanie nawiązane, wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2009.

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt jest uchwytem komunikatu, dla którego wymagany jest bufor.

Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

MHBOPT (MQMHBO)-wejście

Struktura MQMHBO umożliwia aplikacjom określanie opcji sterujących sposobem tworzenia buforów z uchwytów komunikatów.

Szczegółowe informacje można znaleźć w sekcji “MQBMHO (opcje od buforu do uchwytu komunikatu) w systemie IBM i” na stronie 1045.

PRNAME (MQCHARV)-wejście

Nazwa właściwości do umieszczenia w buforze.

Jeśli nie można znaleźć właściwości zgodnej z nazwą, wywołanie kończy się niepowodzeniem z kodem powrotu RC2471.

Znaki wieloznaczne

Można użyć znaku wieloznacznego, aby umieścić więcej niż jedną właściwość w buforze. W tym celu należy użyć znaku procentu (%) na końcu nazwy właściwości. Ten znak wieloznaczny zastępuje zero lub więcej znaków, w tym znak kropki (.).

Więcej informacji na temat używania nazw właściwości zawiera sekcja Nazwy właściwości i sekcja Ograniczenia dotyczące nazw właściwości .

MSGDSC (MQMD)-wejście/wyjście

Struktura *MSGDSC* opisuje zawartość obszaru buforu.

W danych wyjściowych pola *Encoding*, *CodedCharSetId* i *Format* są ustawione w taki sposób, aby poprawnie opisywały kodowanie, identyfikator zestawu znaków i format danych w obszarze buforu zapisywanych przez wywołanie.

Dane w tej strukturze są w zestawie znaków i kodowaniu aplikacji.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

BUFFLEN to długość obszaru buforu w bajtach.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście/wyjście

BUFFER definiuje obszar zawierający bufor komunikatów. W przypadku większości danych należy wyrównać bufor do granicy 4-bajtowej.

Jeśli plik *BUFFER* zawiera dane znakowe lub liczbowe, należy ustawić pola *CodedCharSetId* i *Encoding* w parametrze **MSGDSC** na wartości odpowiednie dla danych. W razie potrzeby umożliwia to konwersję danych.

Jeśli w buforze komunikatów zostaną znalezione właściwości, zostaną one opcjonalnie usunięte. Później staną się dostępne w uchwycie komunikatu po powrocie z wywołania.

W języku programowania C parametr jest zadeklarowany jako wskaźnik-do-void, co oznacza, że jako parametr można podać adres dowolnego typu danych.

Jeśli parametr **BUFLEN** ma wartość zero, oznacza to, że nie występuje odwołanie do parametru *BUFFER* . W takim przypadku adres parametru przekazywany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość NULL.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

DATLEN to długość (w bajtach) właściwości zwróconych w buforze. Jeśli wartość jest równa zero, żadne właściwości nie są zgodne z wartością podaną w pliku *PRNAME* i wywołanie kończy się niepowodzeniem z kodem przyczyny RC2471.

Jeśli wartość *BUFLEN* jest mniejsza niż długość wymagana do zapisania właściwości w buforze, wywołanie MQMHBUF kończy się niepowodzeniem z kodem powrotu RC2469, ale wartość jest nadal wprowadzana do pliku *DATLEN*. Umożliwia to aplikacji określenie wielkości buforu wymaganego do dostosowania właściwości, a następnie ponowne wywołanie z wymaganą wartością *BUFLEN*.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2501
(2501, X'095C') Niepoprawna struktura opcji przesyłania komunikatu do buforu.

RC2004
(2004, X'07D4') Niepoprawny parametr buforu.

RC2005
(2005, X'07D5') Niepoprawny parametr długości buforu.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'07D9') Połączenie z menedżerem kolejek zostało utracone.

RC2010
(2010, X'07DA') Niepoprawny parametr długości danych.

RC2460
(2460, X'099C') Uchwyt komunikatu jest niepoprawny.

RC2026
(2026, X'07EA') Niepoprawny deskryptor komunikatu.

RC2499
(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

RC2442
(2442, X'098A') Nazwa właściwości jest niepoprawna.

RC2471
(2471, X'09A7') Właściwość niedostępna.

RC2469
(2469, X'09A5') Wartość BufferLength jest zbyt mała, aby pomieścić określone właściwości.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQMHBUF(HCONN : HMSG : MHBOPT :
                                        PRNAME : MSGDSC : BUFLEN :
                                        BUFFER : DATLEN :
                                        CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQMHBUF          PR                  EXTPROC('MQMHBUF')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          20I 0 VALUE
D* Options that control the action of MQMHBUF
D MHBOPT                        12A
D* Property name
D PRNAME                        32A
D* Message descriptor
D MSGDSC                       364A
D* Length in bytes of the Buffer area
D BUFLEN                        10I 0 VALUE
D* Area to contain the properties
D BUFFER                          *   VALUE
D* Length of the properties
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQOPEN (Open object) w systemie IBM i

Wywołanie MQOPEN zapewnia dostęp do obiektu.

Poprawne są następujące typy obiektów:

- Kolejka (łącznie z listami dystrybucyjnymi)
- Lista nazw

- Definicja procesu
- Menedżer kolejek
- Temat

Indeks

Składnia

MQOPEN *(HCONN, OBJDSC, OPTS, HOBJ, CMPCOD, REASON)*

Użycie notatek

1. Otwarty obiekt jest jednym z następujących obiektów:
   - Kolejka w celu:
     - Pobieranie lub przeglądanie komunikatów (za pomocą wywołania MQGET)
     - Komunikaty typu put (przy użyciu wywołania MQPUT)
     - Uzyskiwanie informacji o atrybutach kolejki (za pomocą wywołania MQINQ)
     - Ustawianie atrybutów kolejki (za pomocą wywołania MQSET)

     Jeśli kolejka o nazwie jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna.

     Lista dystrybucyjna jest specjalnym typem obiektu kolejki, który zawiera listę kolejek. Można go otwierać w celu umieszczania komunikatów, ale nie w celu pobierania lub przeglądania komunikatów, a także w celu uzyskiwania informacji lub ustawiania atrybutów. Więcej informacji na ten temat zawiera uwaga 8.

     Kolejka o typie `QSGDISP(GROUP)` jest specjalnym typem definicji kolejki, który nie może być używany z wywołaniami MQOPEN lub MQPUT1 .
   - Lista nazw w celu:
     - Sprawdź nazwy kolejek na liście (za pomocą wywołania MQINQ).
   - Definicja procesu w celu:
     - Sprawdź atrybuty procesu (za pomocą wywołania MQINQ).
   - Menedżer kolejek w celu:
     - Sprawdź atrybuty menedżera kolejek lokalnych (przy użyciu wywołania MQINQ).
2. Poprawne jest, aby aplikacja otwierała ten sam obiekt więcej niż jeden raz. Dla każdego otwarcia zwracany jest inny uchwyt obiektu. Każdy zwrócony uchwyt może być użyty dla funkcji, dla których wykonano odpowiednie otwarcie.
3. Jeśli otwierany obiekt jest kolejką, ale nie kolejką klastra, wszystkie tłumaczenia nazw w menedżerze kolejek lokalnych są wykonywane w czasie wywołania MQOPEN. W przypadku konkretnego wywołania MQOPEN może to obejmować jedną lub więcej z następujących opcji:
   - Tłumaczenie aliasu na nazwę kolejki podstawowej
   - Tłumaczenie nazwy lokalnej definicji kolejki zdalnej na nazwę zdalnego menedżera kolejek oraz nazwy, pod jaką kolejka jest znana w zdalnym menedżerze kolejek
   - Tłumaczenie nazwy menedżera kolejek zdalnych na nazwę lokalnej kolejki transmisji

   Należy jednak pamiętać, że kolejne wywołania MQINQ lub MQSET dla uchwytu odnoszą się wyłącznie do nazwy, która została otwarta, a nie do obiektu będącego wynikiem translacji nazw. Na przykład,

jeśli otwarty obiekt jest aliasem, atrybuty zwracane przez wywołanie MQINQ są atrybutami aliasu, a nie atrybutami kolejki podstawowej, na którą alias jest tłumaczony. Sprawdzanie tłumaczenia nazw jest nadal wykonywane, niezależnie od tego, co określono dla parametru **OPTS** w odpowiedniej komendzie MQOPEN.

Jeśli otwierany obiekt jest kolejką klastra, tłumaczenie nazw może wystąpić w czasie wywołania MQOPEN lub może zostać odroczone do czasu późniejszego. Punkt, w którym występuje rozstrzyganie, jest kontrolowany przez opcje OOBND* określone w wywołaniu MQOPEN:

- OBNDO
- OOBNDN
- OOBNDQ

Więcej informacji na temat tłumaczenia nazw dla kolejek klastra zawiera sekcja Rozstrzyganie nazw .

4. Atrybuty obiektu mogą się zmieniać, gdy aplikacja ma otwarty obiekt. W wielu przypadkach aplikacja nie zauważa tego, ale w przypadku niektórych atrybutów menedżer kolejek oznacza uchwyt jako niepoprawny. Są to:
   - Dowolny atrybut, który ma wpływ na tłumaczenie nazw obiektu. Ma to zastosowanie niezależnie od użytych opcji otwierania i obejmuje następujące elementy:
     - Zmiana atrybutu **BaseQName** kolejki aliasowej, która jest otwarta.
     - Zmiana atrybutów kolejki **RemoteQName** lub **RemoteQMgrName** dla dowolnego uchwytu otwartego dla tej kolejki lub dla kolejki, która jest tłumaczona przez tę definicję jako alias menedżera kolejek.
     - Każda zmiana, która powoduje, że aktualnie otwarty uchwyt kolejki zdalnej jest tłumaczony na inną kolejkę *transmisji* lub nie udaje się w ogóle przetłumaczyć na jedną kolejkę. Na przykład może to być:
       - Zmiana atrybutu **XmitQName** lokalnej definicji kolejki zdalnej, bez względu na to, czy definicja jest używana dla kolejki, czy dla aliasu menedżera kolejek.

       Jest od tego jeden wyjątek, a mianowicie utworzenie nowej kolejki transmisji. Uchwyt, który zostałby przetłumaczony na tę kolejkę, gdyby był obecny podczas otwierania uchwytu, ale zamiast tego został przetłumaczony na domyślną kolejkę transmisji, nie jest niepoprawny.
     - Zmiana atrybutu **DefXmitQName** menedżera kolejek. W tym przypadku wszystkie otwarte uchwyty, które zostały przetłumaczone na poprzednio nazwaną kolejkę (które zostały przetłumaczone tylko dlatego, że była to domyślna kolejka transmisji), są oznaczone jako niepoprawne. Nie ma to wpływu na uchwyty, które zostały przetłumaczone na tę kolejkę z innych przyczyn.
   - Atrybut kolejki **Shareability** , jeśli istnieją co najmniej dwa uchwyty, które obecnie zapewniają dostęp OOINPS dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę. Jeśli tak, *wszystkie* uchwyty otwarte dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę, są oznaczane jako niepoprawne, niezależnie od opcji otwarcia.
   - Atrybut kolejki **Usage** dla wszystkich uchwytów otwartych dla tej kolejki lub dla kolejki, która jest tłumaczona na tę kolejkę, niezależnie od opcji otwarcia.

   Jeśli uchwyt jest oznaczony jako niepoprawny, wszystkie kolejne wywołania (inne niż MQCLOSE) używające tego uchwytu kończą się niepowodzeniem z kodem przyczyny RC2041; aplikacja powinna wywołać wywołanie MQCLOSE (przy użyciu oryginalnego uchwytu), a następnie ponownie otworzyć kolejkę. Wszystkie niezatwierdzone aktualizacje dotyczące starego uchwytu z poprzednich pomyślnych wywołań mogą być nadal zatwierdzone lub wycofane, zgodnie z wymaganiami logiki aplikacji.

   Jeśli zmiana atrybutu spowoduje taką zmianę, należy użyć specjalnej wersji komendy "force" .
5. Menedżer kolejek przeprowadza sprawdzanie zabezpieczeń podczas wykonywania wywołania MQOPEN w celu sprawdzenia, czy identyfikator użytkownika, pod którym działa aplikacja, ma odpowiedni poziom uprawnień, zanim dostęp będzie dozwolony. Sprawdzanie uprawnień jest

wykonywane dla nazwy otwieranego obiektu, a nie dla nazwy lub nazw, które są wynikiem translacji nazwy.

Jeśli otwierany obiekt jest kolejką modelową, menedżer kolejek wykonuje pełne sprawdzenie zabezpieczeń zarówno w odniesieniu do nazwy kolejki modelowej, jak i nazwy kolejki dynamicznej, która została utworzona. Jeśli wynikowa kolejka dynamiczna zostanie otwarta jawnie, zostanie wykonane dodatkowe sprawdzenie bezpieczeństwa zasobu względem nazwy kolejki dynamicznej.

6. Kolejkę zdalną można określić na jeden z dwóch sposobów w parametrze **OBJDSC** tego wywołania (patrz opis pól *ODON* i *ODMN* w sekcji "MQOD (deskryptor obiektu) w systemie IBM i" na stronie 1193):

   - Przez określenie dla *ODON* nazwy lokalnej definicji kolejki zdalnej. W tym przypadku parametr *ODMN* odwołuje się do lokalnego menedżera kolejek i może zostać określony jako pusty.

     Sprawdzenie poprawności zabezpieczeń wykonywane przez menedżer kolejek lokalnych sprawdza, czy użytkownik ma uprawnienia do otwarcia lokalnej definicji kolejki zdalnej.
   - Przez określenie dla parametru *ODON* nazwy zdalnej kolejki znanej zdalnemu menedżerowi kolejek. W tym przypadku *ODMN* jest nazwą zdalnego menedżera kolejek.

     Sprawdzanie poprawności zabezpieczeń wykonywane przez menedżer kolejek lokalnych sprawdza, czy użytkownik jest autoryzowany do wysyłania komunikatów do kolejki transmisji w wyniku procesu tłumaczenia nazw.

   W obu przypadkach:

   - Lokalny menedżer kolejek nie wysyła żadnych komunikatów do zdalnego menedżera kolejek w celu sprawdzenia, czy użytkownik ma uprawnienia do umieszczania komunikatów w kolejce.
   - Po nadejściu komunikatu do zdalnego menedżera kolejek zdalny menedżer kolejek może go odrzucić, ponieważ użytkownik, który zainicjuje komunikat, nie jest autoryzowany.
7. Wywołanie MQOPEN z opcją OOBRW ustanawia kursor przeglądania do użycia z wywołaniami MQGET, które określają uchwyt obiektu i jedną z opcji przeglądania. Umożliwia to skanowanie kolejki bez zmiany jej zawartości. Komunikat znaleziony podczas przeglądania można później usunąć z kolejki za pomocą opcji GMMUC.

   Wiele kursorów przeglądania może być aktywnych dla jednej aplikacji, wysyłając kilka żądań MQOPEN dla tej samej kolejki.
8. Poniższe uwagi dotyczą użycia list dystrybucyjnych.

   - Podczas otwierania listy dystrybucyjnej pola w strukturze MQOD muszą być ustawione w następujący sposób:
     - Parametr *ODVER* musi mieć wartość ODVER2 lub większą.
     - *ODOT* musi być równe OTQ.
     - Parametr *ODON* musi być pusty lub musi być łańcuchem o wartości NULL.
     - Parametr *ODMN* musi być pusty lub musi być łańcuchem o wartości NULL.
     - Wartość *ODREC* musi być większa od zera.
     - Jeden z *ODORO* i *ODORP* musi być zerem, a drugi niezerem.
     - Nie więcej niż jedna z wartości *ODRRO* i *ODRRP* może być niezerowa.
     - Muszą istnieć rekordy obiektów *ODREC* , adresowane przez *ODORO* lub *ODORP*. Rekordy obiektów muszą być ustawione na nazwy kolejek docelowych, które mają zostać otwarte.
     - Jeśli jeden z *ODRRO* i *ODRRP* jest niezerowy, muszą istnieć rekordy odpowiedzi *ODREC* . Są one ustawiane przez menedżera kolejek, jeśli wywołanie zostanie zakończone z kodem przyczyny RC2136.

     Za pomocą programu MQOD w wersji version-2 można również otworzyć pojedynczą kolejkę, która nie znajduje się na liście dystrybucyjnej, upewniając się, że parametr *ODREC* ma wartość zero.
   - W parametrze **OPTS** poprawne są tylko następujące opcje otwierania:

- GOTOWE
- OOPAS*
- OOSET*
- OOALTU
- OFIQ

- Kolejki docelowe na liście dystrybucyjnej mogą być kolejkami lokalnymi, aliasowymi lub zdalnymi, ale nie mogą być kolejkami modelowymi. Jeśli określono kolejkę modelową, jej otwarcie nie powiedzie się z kodem przyczyny RC2057. Nie uniemożliwia to jednak pomyślnego otwarcia innych kolejek na liście.
- Parametry kodu zakończenia i kodu przyczyny są ustawione w następujący sposób:
  - Jeśli wszystkie operacje otwarcia kolejek na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, parametry kodu zakończenia i kodu przyczyny są ustawione w taki sposób, aby opisywać wspólny wynik. W tym przypadku rekordy odpowiedzi MQRR (jeśli są udostępniane przez aplikację) nie są ustawione.

    Na przykład, jeśli każde otwarcie powiedzie się, kod zakończenia zostanie ustawiony na CCOK, a kod przyczyny na RCNONE; jeśli każde otwarcie nie powiedzie się, ponieważ żadna z kolejek nie istnieje, parametry zostaną ustawione na CCFAIL i RC2085.
  - Jeśli operacje otwarcia dla kolejek na liście dystrybucyjnej nie powiodą się lub zakończą się niepowodzeniem w ten sam sposób:
    - Parametr kodu zakończenia jest ustawiany na wartość CCWARN, jeśli co najmniej jedno otwarcie zakończyło się pomyślnie, lub na wartość CCFAIL, jeśli wszystkie nie powiodły się.
    - Parametr kodu przyczyny ma wartość RC2136.
    - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.
- Jeśli lista dystrybucyjna została pomyślnie otwarta, uchwyt *HOBJ* zwrócony przez wywołanie może być używany w kolejnych wywołaniach MQPUT do umieszczania komunikatów w kolejkach na liście dystrybucyjnej, a w wywołaniu MQCLOSE do zrzeczenia się dostępu do listy dystrybucyjnej. Jedyną poprawną opcją zamknięcia dla listy dystrybucyjnej jest CONONE.

  Wywołania MQPUT1 można również użyć do umieszczenia komunikatu na liście dystrybucyjnej. Struktura MQOD definiująca kolejki na liście jest określona jako parametr w tym wywołaniu.
- Każde pomyślnie otwarte miejsce docelowe na liście dystrybucyjnej jest traktowane jako *oddzielny* uchwyt podczas sprawdzania, czy aplikacja przekroczyła maksymalną dozwoloną liczbę uchwytów (patrz atrybut menedżera kolejek systemu **MaxHandles**). Ma to miejsce nawet wtedy, gdy co najmniej dwa miejsca docelowe na liście dystrybucyjnej są rzeczywiście tłumaczone na tę samą kolejkę fizyczną. Jeśli wywołanie MQOPEN lub MQPUT1 dla listy dystrybucyjnej spowoduje przekroczenie liczby uchwytów używanych przez aplikację *MaxHandles*, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2017.
- Każde pomyślnie otwarte miejsce docelowe ma wartość atrybutu **OpenOutputCount** zwiększoną o jeden. Jeśli dwa lub więcej miejsc docelowych na liście dystrybucyjnej jest rzeczywiście rozstrzyganych do tej samej kolejki fizycznej, to ta kolejka ma atrybut **OpenOutputCount** zwiększany o liczbę miejsc docelowych na liście dystrybucyjnej, które są rozstrzygane do tej kolejki.
- Każda zmiana definicji kolejek, która spowodowałaby, że uchwyt stałby się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana w ścieżce rozstrzygania), nie powoduje, że uchwyt listy dystrybucyjnej stałby się niepoprawny. Jednak powoduje to niepowodzenie dla tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnym wywołaniu MQPUT.
- Poprawne jest, aby lista dystrybucyjna zawierała tylko jedno miejsce docelowe.

9. Poniższe uwagi dotyczą użycia kolejek klastra.
   - Jeśli kolejka klastra jest otwierana po raz pierwszy, a menedżer kolejek lokalnych nie jest menedżerem kolejek repozytorium pełnego, menedżer kolejek lokalnych uzyskuje informacje o kolejce klastra z menedżera kolejek repozytorium pełnego. Jeśli sieć jest zajęta, odebranie

potrzebnych informacji z menedżera kolejek repozytorium może potrwać kilka sekund przez lokalny menedżer kolejek. W wyniku tego aplikacja wydająca wywołanie MQOPEN może oczekiwać do 10 sekund, zanim sterowanie wróci z wywołania MQOPEN. Jeśli w tym czasie lokalny menedżer kolejek nie otrzyma potrzebnych informacji o kolejce klastra, wywołanie zakończy się niepowodzeniem z kodem przyczyny RC2189.

- Jeśli kolejka klastra jest otwarta, a w klastrze istnieje wiele instancji kolejki, to faktycznie otwarta instancja zależy od opcji określonych w wywołaniu MQOPEN:
  - Jeśli podane opcje obejmują jedną z następujących opcji:
    - OOBRW
    - OOINPQ,
    - OOOINPX
    - OOOINPS
    - OOSET,

    Instancja otwartej kolejki klastra musi być instancją lokalną. Jeśli nie ma lokalnej instancji kolejki, wywołanie MQOPEN nie powiedzie się.
  - Jeśli podane opcje nie zawierają żadnej z powyższych, ale zawierają jedną lub obie z następujących opcji:
    - OOINQ,
    - GOTOWE

    Otwarta instancja jest instancją lokalną (jeśli istnieje), a w przeciwnym razie instancją zdalną. Instancja wybrana przez menedżera kolejek może jednak zostać zmieniona przez wyjście obciążenia klastra (jeśli istnieje).

Więcej informacji na temat kolejek klastrów zawiera sekcja Kolejki klastrów.

10. Do aplikacji uruchamianych przez monitor wyzwalacza jest przekazywana nazwa kolejki, która jest powiązana z aplikacją podczas jej uruchamiania. Tę nazwę kolejki można podać w parametrze **OBJDSC** , aby ją otworzyć. Więcej szczegółów zawiera opis struktury MQTMC.
11. Jeśli używana jest opcja OORLOQ, kolejka lokalna jest już zwracana, gdy otwarta jest kolejka lokalna, aliasowa lub modelowa, ale nie jest to możliwe na przykład wtedy, gdy otwarta jest kolejka zdalna lub nielokalowa kolejka klastrowa; nazwy ResolvedQName i ResolvedQMgrsą wprowadzane z nazwą RemoteQName i nazwą RemoteQMgrznalezioną w definicji kolejki zdalnej lub w podobny sposób z wybraną zdalną kolejką klastrową. Jeśli podczas otwierania kolejki zdalnej podano parametr OORLOQ, na przykład ResolvedQName będzie teraz kolejką transmisji, do której będą umieszczane komunikaty. Zostanie wprowadzona nazwa ResolvedQMgrwraz z nazwą lokalnego menedżera kolejek udostępniającego kolejkę transmisji. Jeśli użytkownik jest uprawniony do przeglądania, wprowadzania lub wyprowadzania danych w kolejce, ma uprawnienia wymagane do określenia tej flagi w wywołaniu MQOPEN. Nie są wymagane żadne uprawnienia specjalne.

Parametry

Wywołanie MQOPEN ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

OBJDSC (MQOD)-wejście/wyjście

Deskryptor obiektu.

Jest to struktura identyfikująca obiekt, który ma zostać otwarty. Szczegółowe informacje na ten temat zawiera sekcja “MQOD (deskryptor obiektu) w systemie IBM i” na stronie 1193 .

Jeśli pole *ODON* w parametrze **OBJDSC** jest nazwą kolejki modelowej, jest to dynamiczna kolejka lokalna. jest tworzona z atrybutami kolejki modelowej; dzieje się tak niezależnie od opcji otwierania określonych przez parametr **OPTS** . Kolejne operacje używające *HOBJ* zwracane przez wywołanie MQOPEN są wykonywane w nowej kolejce dynamicznej, a nie w kolejce modelowej. Dotyczy to również wywołań MQINQ i MQSET. Nazwa kolejki modelowej w parametrze **OBJDSC** jest zastępowana nazwą utworzonej kolejki dynamicznej. Typ kolejki dynamicznej jest określany na podstawie wartości atrybutu **DefinitionType** kolejki modelowej (patrz "Atrybuty kolejek" na stronie 1410). Informacje na temat opcji zamykania mających zastosowanie do kolejek dynamicznych zawiera opis wywołania MQCLOSE.

OPTS (10-cyfrowa liczba całkowita ze znakiem)-wejście

Opcje sterujące działaniem komendy MQOPEN.

Należy podać co najmniej jedną z następujących opcji:

- OOBRW
- OOINP* (tylko jeden z tych)
- OOINQ,
- GOTOWE
- OOSET,
- OORLQ

W razie potrzeby można określić inne opcje. Jeśli wymagana jest więcej niż jedna opcja, wartości mogą zostać dodane (nie należy dodawać tej samej stałej więcej niż raz). Kombinacje, które nie są poprawne, są odnotowywane; wszystkie pozostałe kombinacje są poprawne. Dozwolone są tylko opcje mające zastosowanie do typu obiektu określonego przez parametr *OBJDSC* (patrz sekcja Poprawne opcje MQOPEN dla każdego typu kolejki).

Opcje dostępu: Następujące opcje sterują typem operacji, które mogą być wykonywane na obiekcie:

OOINPQ,
Otwórz kolejkę, aby pobrać komunikaty przy użyciu wartości domyślnej zdefiniowanej przez kolejkę.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Typ dostępu jest współużytkowany lub wyłączny, w zależności od wartości atrybutu kolejki **DefInputOpenOption** . Szczegółowe informacje na ten temat zawiera sekcja "Atrybuty kolejek" na stronie 1410 .

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

OOOINPS
Otwórz kolejkę, aby pobrać komunikaty z dostępem współużytkowanym.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się pomyślnie, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację z wartością OOINPS, ale zakończy się niepowodzeniem z kodem przyczyny RC2042 , jeśli kolejka jest obecnie otwarta z wartością OOINPX.

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

OOOINPX
Otwórz kolejkę, aby pobrać komunikaty z dostępem na wyłączność.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie kończy się niepowodzeniem z kodem przyczyny RC2042 , jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację do wprowadzania danych dowolnego typu (OOINPS lub OOINPX).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami.

Do tych opcji mają zastosowanie następujące uwagi:

- Można podać tylko jedną z tych opcji.
- Wywołanie MQOPEN z jedną z tych opcji może zakończyć się pomyślnie, nawet jeśli atrybut kolejki **InhibitGet** jest ustawiony na wartość QAGETI (chociaż kolejne wywołania MQGET zakończą się niepowodzeniem, gdy atrybut jest ustawiony na tę wartość).
- Jeśli kolejka jest zdefiniowana jako niewspółużytkowalna (czyli atrybut kolejki **Shareability** ma wartość QANSHR), próby otwarcia kolejki dla dostępu współużytkowanego są traktowane jako próby otwarcia kolejki z dostępem na wyłączność.
- Jeśli kolejka aliasowa jest otwarta z jedną z tych opcji, test do wyłącznego użycia (lub do tego, czy inna aplikacja ma wyłączne użycie) jest porównywany z kolejką podstawową, na którą alias jest tłumaczony.
- Te opcje nie są poprawne, jeśli *ODMN* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu **RemoteQMgrName** w lokalnej definicji kolejki zdalnej używanej na potrzeby określania aliasu menedżera kolejek jest nazwą lokalnego menedżera kolejek.

OOBRW
Otwórz kolejkę, aby przeglądać komunikaty.

Kolejka jest otwierana do użycia w kolejnych wywołaniach MQGET z jedną z następujących opcji:

- GMBRWF,
- GMBRWN
- GMBRWC

Jest to dozwolone nawet wtedy, gdy kolejka jest aktualnie otwarta dla OOINPX. Wywołanie MQOPEN z opcją OOBRW ustanawia kursor przeglądania i umieszcza go logicznie przed pierwszym komunikatem w kolejce. Więcej informacji na ten temat zawiera opis pola *GMOPT* w sekcji “MQGMO (opcje pobierania komunikatów) w systemie IBM i” na stronie 1106 .

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Nie jest ona również poprawna, jeśli *ODMN* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu **RemoteQMgrName** w definicji lokalnej kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

GOTOWE
Otwórz kolejkę, aby umieścić komunikaty, lub temat lub łańcuch tematu, aby opublikować komunikaty.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQPUT.

Wywołanie MQOPEN z tą opcją może zakończyć się pomyślnie, nawet jeśli atrybut kolejki **InhibitPut** jest ustawiony na wartość QAPUTI (chociaż kolejne wywołania MQPUT nie powiodą się, gdy atrybut jest ustawiony na tę wartość).

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym list dystrybucyjnych i tematów.

OOINQ,
Otwórz obiekt, aby uzyskać informacje o atrybutach.

Kolejka, lista nazw, definicja procesu lub menedżer kolejek są otwierane do użycia z kolejnymi wywołaniami MQINQ.

Ta opcja jest poprawna dla wszystkich typów obiektów innych niż listy dystrybucyjne. Nie jest on poprawny, jeśli parametr *ODMN* jest nazwą aliasu menedżera kolejek. Jest to prawda, nawet jeśli wartość atrybutu **RemoteQMgrName** w definicji lokalnej kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

OOSET,
Otwórz kolejkę, aby ustawić atrybuty.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQSET.

Ta opcja jest poprawna dla wszystkich typów kolejek innych niż listy dystrybucyjne. Nie jest ona poprawna, jeśli *ODMN* jest nazwą lokalnej definicji kolejki zdalnej. Jest ona poprawna nawet wtedy, gdy wartość atrybutu **RemoteQMgrName** w lokalnej definicji kolejki zdalnej używanej na potrzeby aliasowania menedżera kolejek jest nazwą lokalnego menedżera kolejek.

Opcje powiązania: Następujące opcje mają zastosowanie, gdy otwierany obiekt jest kolejką klastra. Te opcje sterują powiązaniem uchwytu kolejki z instancją kolejki klastra:

OBNDO
Powiąż uchwyt z miejscem docelowym, gdy kolejka jest otwarta.

Spowoduje to, że lokalny menedżer kolejek powiąże uchwyt kolejki z instancją kolejki docelowej po otwarciu kolejki. W rezultacie wszystkie komunikaty umieszczone za pomocą tego uchwytu są wysyłane do tej samej instancji kolejki docelowej i tej samej trasy.

Ta opcja jest poprawna tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja zostanie określona dla kolejki, która nie jest kolejką klastra, opcja zostanie zignorowana.

OOBNDN
Nie wiąż się z konkretnym miejscem docelowym.

Spowoduje to zatrzymanie lokalnego menedżera kolejek, który powiąże uchwyt kolejki z instancją kolejki docelowej. W rezultacie kolejne wywołania MQPUT używające tego uchwytu mogą spowodować wysłanie komunikatów do *różnych* instancji kolejki docelowej lub do tej samej instancji, ale do różnych tras. Pozwala również na późniejszą zmianę wybranej instancji przez lokalny menedżer kolejek, przez zdalny menedżer kolejek lub przez agent kanału komunikatów (MCA) zgodnie z warunkami sieciowymi.

Uwaga: Aplikacje klienckie i serwerowe, które muszą wymieniać *serię* komunikatów w celu zakończenia transakcji, nie powinny używać OOBNDN (lub OOBNDQ, gdy *DefBind* ma wartość BNDNOT), ponieważ kolejne komunikaty w serii mogą być wysyłane do różnych instancji aplikacji serwera.

Jeśli dla kolejki klastra określono opcję OOBRW lub jedną z opcji OOINP*, menedżer kolejek musi wybrać lokalną instancję kolejki klastra. W wyniku tego powiązanie uchwytu kolejki jest stałe, nawet jeśli określono parametr OOBNDN.

Jeśli wartość OOINQ jest określona z wartością OOBNDN, kolejne wywołania MQINQ używające tego uchwytu mogą prowadzić zapytania do różnych instancji kolejki klastra, chociaż zwykle wszystkie instancje mają takie same wartości atrybutów.

Parametr OOBNDN jest poprawny tylko dla kolejek i dotyczy tylko kolejek klastra. Jeśli ta opcja zostanie określona dla kolejki, która nie jest kolejką klastra, opcja zostanie zignorowana.

OOBNDQ
Użyj domyślnego powiązania dla kolejki.

Powoduje to, że lokalny menedżer kolejek wiąże uchwyt kolejki w sposób zdefiniowany przez atrybut kolejki **DefBind** . Wartością tego atrybutu jest BNDOPN lub BNDNOT.

OOBNDQ jest wartością domyślną, jeśli nie określono OOBNDO i OOBNDN.

OOBNDQ jest zdefiniowany w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną z pozostałych dwóch opcji wiązania, ale ponieważ jej wartość wynosi zero, nie można jej wykryć.

Opcje kontekstu: Następujące opcje sterują przetwarzaniem kontekstu komunikatu:

OOSAVA
Zapisz kontekst po pobraniu komunikatu.

Informacje o kontekście są powiązane z tym uchwytem kolejki. Te informacje są ustawiane na podstawie kontekstu dowolnego komunikatu pobranego za pomocą tego uchwytu. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Te informacje o kontekście można przekazać do komunikatu umieszczanego później w kolejce przy użyciu wywołań MQPUT lub MQPUT1 . Patrz opcje PMPASI i PMPASA opisane w sekcji “MQPMO (opcje umieszczania komunikatów) w systemie IBM i” na stronie 1208.

Dopóki komunikat nie zostanie pomyślnie pobrany, nie można przekazać kontekstu do komunikatu umieszczanego w kolejce.

Komunikat pobrany za pomocą jednej z opcji przeglądania GMBRW* nie ma zapisanych informacji o kontekście (chociaż pola kontekstu w parametrze **MSGDSC** są ustawiane po przeglądaniu).

Ta opcja jest poprawna tylko dla kolejek lokalnych, aliasowych i modelowych; nie jest poprawna dla kolejek zdalnych, list dystrybucyjnych i obiektów, które nie są kolejkami. Należy podać jedną z opcji OOINP*.

OOPASI
Zezwalaj na przekazywanie kontekstu tożsamości.

Umożliwia to określenie opcji PMPASI w parametrze **PMO** , gdy komunikat jest umieszczany w kolejce. W ten sposób komunikat otrzymuje informacje o kontekście tożsamości z kolejki wejściowej, która została otwarta za pomocą opcji OOSAVA. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Należy podać opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

OOPASA
Zezwalaj na przekazywanie całego kontekstu.

Pozwala to na określenie opcji PMPASA w parametrze **PMO** , gdy komunikat jest umieszczany w kolejce. Daje to komunikatowi informacje o tożsamości i kontekście źródłowym z kolejki wejściowej, która została otwarta za pomocą opcji OOSAVA. Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Ta opcja implikuje działanie OOPASI, które nie musi być określone. Należy podać opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

OOSETI
Zezwalaj na ustawianie kontekstu tożsamości.

Umożliwia to określenie opcji PMSETI w parametrze **PMO** , gdy komunikat jest umieszczany w kolejce. Daje to komunikatowi informacje o kontekście tożsamości zawarte w parametrze **MSGDSC** określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Ta opcja implikuje działanie OOPASI, które nie musi być określone. Należy podać opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

OOSETA
Zezwalaj na ustawienie całego kontekstu.

Umożliwia to określenie opcji PMSETA w parametrze **PMO** , gdy komunikat jest umieszczany w kolejce. Daje to komunikatowi informacje o tożsamości i kontekście źródłowym zawarte w parametrze **MSGDSC** określonym w wywołaniu MQPUT lub MQPUT1 . Więcej informacji na temat kontekstu komunikatu zawiera sekcja Kontekst komunikatu i sekcja Sterowanie kontekstem.

Ta opcja oznacza następujące opcje, które nie muszą być określone:

- OOPASI
- OOPASA
- OOSETI

Należy podać opcję OOOUT.

Ta opcja jest poprawna dla wszystkich typów kolejek, w tym dla list dystrybucyjnych.

Inne opcje: Następujące opcje sterują sprawdzaniem autoryzacji i tym, co się dzieje, gdy menedżer kolejek jest wyciszany:

OOALTU
Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

Wskazuje to, że pole *ODAU* w parametrze **OBJDSC** zawiera identyfikator użytkownika, który ma być używany do sprawdzania poprawności tego wywołania MQOPEN. Wywołanie może zakończyć się powodzeniem tylko wtedy, gdy *ODAU* ma uprawnienia do otwarcia obiektu z określonymi opcjami dostępu, bez względu na to, czy identyfikator użytkownika, pod którym działa aplikacja, jest do tego uprawniony. Nie ma to jednak zastosowania do określonych opcji kontekstu, które są zawsze sprawdzane pod kątem identyfikatora użytkownika, pod którym działa aplikacja.

Ta opcja jest poprawna dla wszystkich typów obiektów.

OFIQ
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

Ta opcja wymusza niepowodzenie wywołania MQOPEN, jeśli menedżer kolejek jest w stanie wyciszania.

Ta opcja jest poprawna dla wszystkich typów obiektów.

OORLQ
Wprowadź nazwę kolejki lokalnej, która została otwarta.

Ta opcja określa, że nazwa ResolvedQName w strukturze MQOD (jeśli jest dostępna) powinna zostać wprowadzona z nazwą kolejki lokalnej, która została otwarta. Podobnie zostanie wprowadzona nazwa ResolvedQMgrz nazwą lokalnego menedżera kolejek udostępniającego kolejkę lokalną.

Tabela 750. Poprawne opcje MQOPEN dla każdego typu kolejki

Opcja	Alias ("1" na stronie 1373)	Lokalne i modelowe	Zdalna	Klaster nielokalny	Lista dystrybucyjna	Temat
OOINPQ,	✓	✓	-	-	-	-
OOOINPS	✓	✓	-	-	-	-
OOOINPX	✓	✓	-	-	-	-
OOBRW	✓	✓	-	-	-	-
GOTOWE	✓	✓	✓	✓	✓	✓
OOINQ,	✓	✓	"2" na stronie 1373	✓	-	-
OOSET,	✓	✓	"2" na stronie 1373	-	-	-
OOBNDO ("3" na stronie 1373)	✓	✓	✓	✓	✓	-
OOBNDN ("3" na stronie 1373)	✓	✓	✓	✓	✓	-
OOBNDQ ("3" na stronie 1373)	✓	✓	✓	✓	✓	-

Tabela 750. Poprawne opcje MQOPEN dla każdego typu kolejki (kontynuacja)						
Opcja	**Alias ("1" na stronie 1373)**	**Lokalne i modelowe**	**Zdalna**	**Klaster nielokalny**	**Lista dystrybucyjna**	**Temat**
OOSAVA	✓	✓	-	-	-	-
OOPASI	✓	✓	✓	✓	✓	"5" na stronie 1373
OOPASA	✓	✓	✓	✓	✓	"5" na stronie 1373
OOSETI	✓	✓	✓	✓	✓	"5" na stronie 1373
OOSETA	✓	✓	✓	✓	✓	"5" na stronie 1373
OOALTU	✓	✓	✓	✓	✓	✓
OFIQ	✓	✓	✓	✓	✓	✓
OORLQ	✓	✓	✓	✓	-	-

Uwagi:

1. Poprawność opcji dla aliasów zależy od poprawności opcji dla kolejki, na którą alias jest tłumaczony.
2. Ta opcja jest poprawna tylko dla lokalnej definicji kolejki zdalnej.
3. Tę opcję można określić dla dowolnego typu kolejki, ale jest ona ignorowana, jeśli kolejka nie jest kolejką klastra.
4. Ten atrybut jest ignorowany w przypadku tematu.
5. Te atrybuty mogą być używane z tematem, ale mają wpływ tylko na kontekst ustawiony dla zachowanego komunikatu, a nie na pola kontekstu wysyłane do dowolnego subskrybenta.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Uchwyt obiektu.

Ten uchwyt reprezentuje dostęp, który został ustanowiony dla obiektu. Musi być określona w kolejnych wywołaniach kolejkowania komunikatów, które działają na obiekcie. Przestaje być ona poprawna po wywołaniu MQCLOSE lub po zakończeniu jednostki przetwarzania definiującej zasięg uchwytu.

Zakres uchwytu jest ograniczony do najmniejszej jednostki Przetwarzanie równoległe obsługiwane przez platformę, na której działa aplikacja. Uchwyt nie jest poprawny poza jednostką przetwarzania równoległego, z której wywołano wywołanie MQOPEN:

- W systemie IBM izasięgiem uchwytu jest zadanie wywołujące wywołanie.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

Deklaracja RPG

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                     CALLP     MQOPEN(HCONN : OBJDSC : OPTS :
     C                                      HOBJ : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQOPEN           PR                  EXTPROC('MQOPEN')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object descriptor
D OBJDSC                        468A
D* Options that control the action of MQOPEN
D OPTS                           10I 0 VALUE
D* Object handle
D HOBJ                           10I 0
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
```

IBM i MQPUT (umieszczaj komunikat) w systemie IBM i

Wywołanie MQPUT umieszcza komunikat w kolejce, na liście dystrybucyjnej lub w temacie. Kolejka, lista dystrybucyjna lub temat muszą być już otwarte.

Składnia

MQPUT *(HCONN, HOBJ, MSGDSC, PMO, BUFLEN, BUFFER, CMPCOD, REASON)*

Użycie notatek

Tematy

Do korzystania z tematów mają zastosowanie następujące uwagi:

1. W przypadku używania MQPUT do publikowania komunikatów w temacie, w którym jeden lub więcej subskrybentów tego tematu nie może otrzymać publikacji z powodu problemu z ich kolejką subskrybentów (na przykład jest pełna), kod przyczyny zwracany do wywołania MQPUT i zachowanie dostarczania jest zależne od ustawienia atrybutów PMSGDLV lub NPMSGDLV w temacie TOPIC. Należy zauważyć, że dostarczenie publikacji do kolejki niedostarczonych komunikatów, gdy określona jest wartość RODLQ, lub usunięcie komunikatu, gdy określona jest wartość RODISC, jest uważane za pomyślne dostarczenie komunikatu. Jeśli żadna z publikacji nie zostanie dostarczona, komenda MQPUT zwróci kod RC2502. Może to wystąpić w następujących przypadkach:

   - Komunikat jest publikowany w temacie TOPIC z parametrem PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na wartość ALL, a każda subskrypcja (trwała lub nie) ma kolejkę, która nie może odebrać publikacji.
   - Komunikat jest publikowany w temacie TOPIC z PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALLDUR, a trwała subskrypcja ma kolejkę, która nie może odebrać publikacji.

   Komenda MQPUT może zwracać wartość z wartością RCNONE, nawet jeśli publikacje nie mogą być dostarczane do niektórych subskrybentów w następujących przypadkach:

   - Komunikat jest publikowany w temacie TOPIC z PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALLAVAIL, a każda subskrypcja, trwała lub nie, ma kolejkę, która nie może odebrać publikacji.
   - Komunikat jest publikowany w temacie TOPIC z PMSGDLV lub NPMSGDLV (w zależności od trwałości komunikatu) ustawionym na ALLDUR, a nietrwała subskrypcja ma kolejkę, która nie może odebrać publikacji.
2. Jeśli nie ma subskrybentów używanego tematu, opublikowany komunikat nie jest wysyłany do żadnej kolejki i jest odrzucany. Nie ma znaczenia, czy ten komunikat jest trwały, czy nietrwały, czy też ma nieograniczony czas utraty ważności lub krótki czas utraty ważności. Jeśli nie ma subskrybentów, komunikat jest nadal usuwany. Wyjątkiem jest sytuacja, w której komunikat ma zostać zachowany. W takim przypadku, mimo że komunikat nie jest wysyłany do kolejek subskrybentów, jest on zapisywany w temacie, który ma zostać dostarczony do nowych subskrypcji lub do subskrybentów, którzy proszą o zachowane publikacje przy użyciu komendy MQSUBRQ.

MQPUT i MQPUT1

Zarówno wywołania MQPUT, jak i MQPUT1 mogą być używane do umieszczania komunikatów w kolejce. Wywołanie, które ma być używane, zależy od okoliczności.

- Wywołania MQPUT należy używać, gdy wiele komunikatów ma być umieszczonych w *tej samej* kolejce.

  Najpierw wysyłane jest wywołanie MQOPEN określające opcję OOOUT, a następnie jedno lub więcej żądań MQPUT w celu dodania komunikatów do kolejki; na końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1 .
- Wywołania MQPUT1 należy używać, gdy tylko *jeden* komunikat ma zostać umieszczony w kolejce.

  To wywołanie hermetyzuje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wykonane.

Kolejki docelowe

Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez użycia grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są następujące warunki. Niektóre warunki dotyczą zarówno lokalnej, jak i zdalnej kolejki docelowej; inne dotyczą tylko zdalnych kolejek docelowych.

Warunki dla lokalnych i zdalnych kolejek docelowych

- Wszystkie wywołania MQPUT znajdują się w tej samej jednostce pracy lub żadne z nich nie znajduje się w jednostce pracy.

  Jeśli komunikaty są umieszczane w określonej kolejce w ramach pojedynczej jednostki pracy, komunikaty z innych aplikacji mogą być przeplatane sekwencją komunikatów w kolejce.

- Wszystkie wywołania MQPUT są wykonywane przy użyciu tego samego uchwytu obiektu *HOBJ*.

  W niektórych środowiskach sekwencja komunikatów jest również zachowywana, gdy używane są różne uchwyty obiektów, pod warunkiem, że wywołania są wykonywane z tej samej aplikacji. Znaczenie "tej samej aplikacji" jest określane przez środowisko:

  – W systemie IBM iaplikacją jest zadanie.
- Wszystkie komunikaty mają ten sam priorytet.

Dodatkowe warunki dla zdalnych kolejek docelowych

- Istnieje tylko jedna ścieżka od wysyłającego menedżera kolejek do docelowego menedżera kolejek.

  Jeśli istnieje możliwość, że niektóre komunikaty w sekwencji mogą znajdować się w innej ścieżce (na przykład z powodu rekonfiguracji, równoważenia ruchu lub wyboru ścieżki na podstawie wielkości komunikatu), kolejność komunikatów w menedżerze kolejek docelowych nie może być zagwarantowana.
- Komunikaty nie są tymczasowo umieszczane w kolejkach niedostarczonych komunikatów w nadawczych, pośrednich lub docelowych menedżerach kolejek.

  Jeśli co najmniej jeden komunikat zostanie tymczasowo umieszczony w kolejce niedostarczonych komunikatów (na przykład z powodu tymczasowego zapełnienia kolejki transmisji lub kolejki docelowej), komunikaty mogą dotrzeć do kolejki docelowej poza kolejnością.
- Wszystkie komunikaty są trwałe lub nietrwałe.

  Jeśli kanał na trasie między nadawczym i docelowym menedżerem kolejek ma atrybut **CDNPM** ustawiony na wartość NPFAST, komunikaty nietrwałe mogą być przesyłane przed komunikatami trwałymi, co powoduje, że kolejność komunikatów trwałych względem komunikatów nietrwałych nie jest zachowywana. Zachowywana jest jednak kolejność komunikatów trwałych względem siebie i komunikatów nietrwałych względem siebie.

Jeśli te warunki nie są spełnione, można użyć grup komunikatów do zachowania kolejności komunikatów, ale należy zauważyć, że wymaga to od aplikacji wysyłających i odbierających użycia obsługi grupowania komunikatów. Więcej informacji na temat grup komunikatów zawiera sekcja:

- Pole *MDMFL* w strukturze MQMD
- Opcja PMLOGO w MQPMO
- Opcja GMLOGO w MQGMO

Lista dystrybucyjna

Poniższe uwagi dotyczą użycia list dystrybucyjnych.

1. Komunikaty mogą być umieszczane na liście dystrybucyjnej przy użyciu programu MQPMO w wersji version-1 lub version-2 . Jeśli używana jest usługa MQPMO w wersji version-1 (lub usługa MQPMO w wersji version-2 z wartością *PMREC* równą zero), aplikacja nie może dostarczyć żadnych rekordów komunikatów umieszczania ani rekordów odpowiedzi. Oznacza to, że nie będzie możliwe zidentyfikowanie kolejek, w których wystąpiły błędy, jeśli komunikat zostanie pomyślnie wysłany do niektórych kolejek na liście dystrybucyjnej, a nie do innych.

   Jeśli rekordy komunikatów umieszczania lub rekordy odpowiedzi są udostępniane przez aplikację, pole *PMVER* musi być ustawione na wartość PMVER2.

   Funkcja MQPMO w wersji version-2 może być również używana do wysyłania komunikatów do pojedynczej kolejki, która nie znajduje się na liście dystrybucyjnej. W tym celu należy upewnić się, że parametr *PMREC* ma wartość zero.
2. Parametry kodu zakończenia i kodu przyczyny są ustawione w następujący sposób:

   - Jeśli wszystkie operacje umieszczania w kolejkach na liście dystrybucyjnej powiodą się lub zakończą się niepowodzeniem w ten sam sposób, kod zakończenia i parametry kodu przyczyny są ustawiane w celu opisania wspólnego wyniku. W tym przypadku rekordy odpowiedzi MQRR (jeśli są udostępniane przez aplikację) nie są ustawione.

Na przykład, jeśli każde umieszczenie powiedzie się pomyślnie, kod zakończenia jest ustawiany na wartość CCOK, a kod przyczyny na wartość RCNONE; jeśli każde umieszczenie nie powiedzie się z powodu zablokowania wszystkich kolejek dla operacji put, parametry są ustawiane na wartości CCFAIL i RC2051.

- Jeśli operacje umieszczania w kolejkach na liście dystrybucyjnej nie powiodą się lub zakończą się niepowodzeniem w ten sam sposób:
  - Parametr kodu zakończenia jest ustawiany na wartość CCWARN, jeśli co najmniej jedno umieszczenie powiodło się, lub na wartość CCFAIL, jeśli wszystkie nie powiodły się.
  - Parametr kodu przyczyny ma wartość RC2136.
  - Rekordy odpowiedzi (jeśli są udostępniane przez aplikację) są ustawiane na indywidualne kody zakończenia i kody przyczyny dla kolejek na liście dystrybucyjnej.

  Jeśli operacja put dla miejsca docelowego nie powiedzie się z powodu niepowodzenia operacji open dla tego miejsca docelowego, pola w rekordzie odpowiedzi są ustawiane na wartości CCFAIL i RC2137; to miejsce docelowe jest uwzględniane w pliku *PMIDC*.

3. Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę lokalną, komunikat jest umieszczany w tej kolejce w normalnej formie (nie jako komunikat listy dystrybucyjnej). Jeśli więcej niż jedno miejsce docelowe jest tłumaczone na tę samą kolejkę lokalną, jeden komunikat jest umieszczany w kolejce dla każdego takiego miejsca docelowego.

   Jeśli miejsce docelowe na liście dystrybucyjnej jest tłumaczone na kolejkę zdalną, komunikat jest umieszczany w odpowiedniej kolejce transmisji. W przypadku gdy kilka miejsc docelowych jest rozstrzyganych do tej samej kolejki transmisji, w kolejce transmisji może zostać umieszczony pojedynczy komunikat listy dystrybucyjnej zawierający te miejsca docelowe, nawet jeśli miejsca te nie znajdowały się obok siebie na liście miejsc docelowych udostępnionej przez aplikację. Można to jednak zrobić tylko wtedy, gdy kolejka transmisji obsługuje komunikaty listy dystrybucyjnej (patrz atrybut kolejki **DistLists** opisany w sekcji “Atrybuty kolejek” na stronie 1410).

   Jeśli kolejka transmisji nie obsługuje list dystrybucyjnych, jedna kopia komunikatu w normalnej formie jest umieszczana w kolejce transmisji dla każdego miejsca docelowego używającego tej kolejki transmisji.

   Jeśli lista dystrybucyjna z danymi komunikatu aplikacji jest zbyt duża dla kolejki transmisji, komunikat listy dystrybucyjnej jest dzielony na mniejsze komunikaty listy dystrybucyjnej, z których każdy zawiera mniej miejsc docelowych. Jeśli dane komunikatu aplikacji mieszczą się tylko w kolejce, nie można w ogóle używać komunikatów listy dystrybucyjnej, a menedżer kolejek generuje jedną kopię komunikatu w normalnej formie dla każdego miejsca docelowego używającego tej kolejki transmisji.

   Jeśli różne miejsca docelowe mają inny priorytet lub trwałość komunikatu (taka sytuacja może wystąpić, gdy aplikacja określa PRQDEF lub PEQDEF), komunikaty nie są wstrzymywane w tym samym komunikacie listy dystrybucyjnej. Zamiast tego menedżer kolejek generuje tyle komunikatów listy dystrybucyjnej, ile jest potrzebnych do obsługi różnych wartości priorytetu i trwałości.

4. Umieszczenie na liście dystrybucyjnej może spowodować:
   - Pojedynczy komunikat listy dystrybucyjnej lub
   - Liczba mniejszych komunikatów z listy dystrybucyjnej lub
   - Połączenie komunikatów listy dystrybucyjnej i komunikatów normalnych, lub
   - Tylko normalne komunikaty.

   To, które z powyższych zdarzeń ma miejsce, zależy od tego, czy:
   - Miejsca docelowe na liście są lokalne, zdalne lub mieszane.
   - Miejsca docelowe mają taki sam priorytet i trwałość komunikatu.
   - Kolejki transmisji mogą przechowywać komunikaty listy dystrybucyjnej.
   - Maksymalna długość komunikatów kolejki transmisji jest wystarczająco duża, aby pomieścić komunikat w formie listy dystrybucyjnej.

Jednak niezależnie od tego, która z powyższych sytuacji ma miejsce, każdy komunikat *fizyczny* będący wynikiem (to znaczy każdy normalny komunikat lub komunikat z listy dystrybucyjnej wynikający z operacji umieszczania) jest liczony jako *jeden* komunikat, gdy:

- Sprawdzanie, czy aplikacja przekroczyła maksymalną dozwoloną liczbę komunikatów w jednostce pracy (patrz atrybut menedżera kolejek systemu **MaxUncommittedMsgs**).
- Sprawdzanie, czy warunki wyzwalania są spełnione.
- Zwiększanie głębokości kolejki i sprawdzanie, czy zostanie przekroczone maksymalne zapełnienie kolejki.

5. Każda zmiana definicji kolejek, która spowodowałaby, że uchwyt stałby się niepoprawny, gdyby kolejki były otwierane indywidualnie (na przykład zmiana w ścieżce rozstrzygania), nie powoduje, że uchwyt listy dystrybucyjnej stałby się niepoprawny. Jednak powoduje to niepowodzenie dla tej konkretnej kolejki, gdy uchwyt listy dystrybucyjnej jest używany w kolejnym wywołaniu MQPUT.

Nagłówki

Jeśli komunikat jest umieszczany z jedną lub większą liczbą struktur nagłówka IBM MQ na początku danych komunikatu aplikacji, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka, aby sprawdzić, czy są one poprawne. Jeśli menedżer kolejek wykryje błąd, wywołanie zakończy się niepowodzeniem z odpowiednim kodem przyczyny. Wykonywane kontrole różnią się w zależności od konkretnych struktur, które są obecne. Ponadto sprawdzenia są wykonywane tylko wtedy, gdy w wywołaniu MQPUT lub MQPUT1 używany jest deskryptor MQMD w wersji version-2 lub nowszej; sprawdzenia nie są wykonywane, jeśli używany jest deskryptor MQMD w wersji version-1 , nawet jeśli na początku danych komunikatu aplikacji znajduje się deskryptor MQMDE.

Następujące struktury nagłówków IBM MQ są w pełni sprawdzone przez menedżer kolejek: MQDH, MQMDE.

W przypadku innych struktur nagłówków IBM MQ menedżer kolejek wykonuje pewne sprawdzanie poprawności, ale nie sprawdza każdego pola. Nie jest sprawdzana poprawność struktur, które nie są obsługiwane przez menedżer kolejek lokalnych, oraz struktur następujących po pierwszym MQDLH w komunikacie.

Oprócz ogólnych kontroli pól w strukturach IBM MQ muszą być spełnione następujące warunki:

- Struktura IBM MQ nie może być podzielona na dwa lub więcej segmentów-struktura musi być całkowicie zawarta w jednym segmencie.
- Suma długości struktur w komunikacie PCF musi być równa długości określonej przez parametr **BUFLEN** w wywołaniu MQPUT lub MQPUT1 . Komunikat PCF to komunikat, który ma jedną z następujących nazw formatu:
  - FMADMN
  - FMEVNT,
  - FMPCF,
- Struktury IBM MQ nie mogą być obcinane, z wyjątkiem następujących sytuacji, w których obcinane struktury są dozwolone:
  - Komunikaty, które są komunikatami raportów.
  - Komunikaty PCF.
  - Komunikaty zawierające strukturę MQDLH. (Struktury *następujące* po pierwszym MQDLH mogą zostać obcięte; struktury poprzedzające MQDLH nie mogą zostać obcięte).

Buforuj

Parametr **BUFFER** przedstawiony w przykładzie programowania w języku RPG jest zadeklarowany jako łańcuch, co ogranicza maksymalną długość parametru do 256 bajtów. Jeśli wymagany jest większy bufor, parametr powinien być zadeklarowany jako struktura lub jako pole w zbiorze fizycznym. Spowoduje to zwiększenie maksymalnej możliwej długości do około 32 kB.

Parametry

Wywołanie MQPUT ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje kolejkę, do której komunikat jest dodawany, lub temat, do którego komunikat jest publikowany. Wartość *HOBJ* została zwrócona przez poprzednie wywołanie MQOPEN, które określało opcję OOOUT.

MSGDSC (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Ta struktura opisuje atrybuty wysyłanego komunikatu i odbiera informacje o komunikacie po zakończeniu żądania umieszczenia. Szczegółowe informacje można znaleźć w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.

Jeśli aplikacja udostępnia strukturę MQMD version-1 , dane komunikatu mogą być poprzedzone strukturą MQMDE w celu określenia wartości pól, które istnieją w strukturze MQMD version-2 , ale nie w strukturze version-1. Pole *MDFMT* w strukturze MQMD musi być ustawione na wartość FMMDE, aby wskazać, że istnieje struktura MQMDE. Więcej informacji na temat zawiera sekcja “MQMDE (Rozszerzenie deskryptora komunikatu) w systemie IBM i” na stronie 1187.

PMO (MQPMO)-wejście/wyjście

Opcje sterujące działaniem komendy MQPUT.

Szczegółowe informacje można znaleźć w sekcji “MQPMO (opcje umieszczania komunikatów) w systemie IBM i” na stronie 1208.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość komunikatu w pliku *BUFFER*.

Zero jest poprawne i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit dla *BUFLEN* zależy od różnych czynników:

- Jeśli kolejka docelowa jest kolejką współużytkowaną, górny limit wynosi 63 kB (64 512 bajtów).
- Jeśli miejsce docelowe jest kolejką lokalną lub jest rozstrzygane jako kolejka lokalna (ale nie jest kolejką współużytkowaną), górny limit zależy od tego, czy:
  - Lokalny menedżer kolejek obsługuje segmentację.
  - Aplikacja wysyłająca określa flagę, która umożliwia menedżerowi kolejek segmentowanie komunikatu. Ta opcja ma wartość MFSEGA i można ją określić w deskryptorze MQMD version-2 lub w deskryptorze MQMDE używanym z deskryptorze MQMD version-1 .

  Jeśli oba te warunki są spełnione, wartość *BUFLEN* nie może przekroczyć 999 999 999 minus wartość pola *MDOFF* w strukturze MQMD. Najdłuższy komunikat logiczny, który można umieścić, to 999 999 999 bajtów (jeśli parametr *MDOFF* ma wartość zero). Jednak ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym działa aplikacja, mogą spowodować obniżenie limitu.

  Jeśli jeden lub oba opisane wcześniej warunki nie są spełnione, produkt *BUFLEN* nie może przekroczyć mniejszej wartości atrybutu **MaxMsgLength** kolejki i atrybutu **MaxMsgLength** menedżera kolejek.

- Jeśli miejsce docelowe jest kolejką zdalną lub jest rozstrzygane na kolejkę zdalną, mają zastosowanie warunki dla kolejek lokalnych *, ale w każdym menedżerze kolejek, przez który komunikat musi przejść, aby osiągnąć kolejkę docelową* . w szczególności:

  1. Lokalna kolejka transmisji używana do tymczasowego przechowywania komunikatu w menedżerze kolejek lokalnych
  2. Pośrednie kolejki transmisji (jeśli istnieją) używane do przechowywania komunikatu w menedżerach kolejek na trasie między lokalnymi i docelowymi menedżerami kolejek
  3. Kolejka docelowa w menedżerze kolejek docelowych

  Najdłuższy komunikat, który można umieścić, jest więc zarządzany przez najbardziej restrykcyjne z tych kolejek i menedżerów kolejek.

  Gdy komunikat znajduje się w kolejce transmisji, dodatkowe informacje rezydują z danymi komunikatu, co zmniejsza ilość danych aplikacji, które mogą być przenoszone. W tej sytuacji podczas określania limitu dla parametru *BUFLEN*zaleca się odjęcie bajtów LNMHD od wartości *MaxMsgLength* kolejek transmisji.

  Uwaga: Tylko niespełnienie warunku 1 może być diagnozowane synchronicznie (z kodem przyczyny RC2030 lub RC2031) podczas umieszczania komunikatu. Jeśli warunki 2 lub 3 nie są spełnione, komunikat jest przekierowywany do kolejki niedostarczonych komunikatów (niedostarczonych komunikatów) w pośrednim menedżerze kolejek lub w docelowym menedżerze kolejek. W takim przypadku generowany jest komunikat raportu, jeśli został on zażądany przez nadawcę.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście

Dane komunikatu.

Jest to bufor zawierający dane aplikacji do wysłania. Bufor powinien być wyrównany do granicy odpowiedniej dla rodzaju danych w komunikacie. Wyrównanie 4-bajtowe powinno być odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówków MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli plik *BUFFER* zawiera dane znakowe, dane liczbowe lub oba te elementy, pola *MDCSI* i *MDENC* w parametrze **MSGDSC** powinny być ustawione na wartości odpowiednie dla danych; umożliwi to odbiorcy komunikatu konwersję danych (w razie potrzeby) na zestaw znaków i kodowanie używane przez odbiornik.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** oraz w kodowaniu lokalnego menedżera kolejek określonym przez translację ENNAT.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2104
(2104, X'838 ') Opcja raportu w deskryptorze komunikatu nie została rozpoznana.

RC2136
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

Jeśli *CMPCOD* to CCFAIL:

RC2004
(2004, X'7D4') Niepoprawny parametr buforu.

RC2005
(2005, X'7D5') Niepoprawny parametr długości buforu.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2013
(2013, X'7DD') Niepoprawny czas ważności.

RC2014
(2014, X'7DE') Kod zapisu czynności jest niepoprawny.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2024
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

RC2026
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

RC2027
(2027, X'7EB') Brak kolejki odpowiedzi.

RC2029
(2029, X'7ED') Niepoprawny typ komunikatu w deskryptorze komunikatu.

RC2030
(2030, X'7EE') Długość komunikatu przekracza maksimum dla kolejki.

RC2031
(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

RC2039
(2039, X'7F7') Kolejka nie jest otwarta do wyprowadzania.

RC2041
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2047
(2047, X'7FF') Trwałość jest niepoprawna.

RC2048
(2048, X'800 ') Kolejka nie obsługuje komunikatów trwałych.

RC2050
(2050, X'802 ') Priorytet komunikatu jest niepoprawny.

RC2051
(2051, X'803 ') Wywołania umieszczania zablokowane dla kolejki.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2053
(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

RC2056
(2056, X'808 ') Brak miejsca na dysku dla kolejki.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2061
(2061, X'80D') Opcje raportu w deskryptorze komunikatu są niepoprawne.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2072
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2093
(2093, X'82D') Kolejka nie jest otwarta dla przekazywania całego kontekstu.

RC2094
(2094, X'82E') Kolejka nie jest otwarta dla przekazywania kontekstu tożsamości.

RC2095
(2095, X'82F') Kolejka nie jest otwarta dla ustawiania całego kontekstu.

RC2096
(2096, X'830 ') Kolejka nie jest otwarta dla ustawionego kontekstu tożsamości.

RC2097
(2097, X'831 ') Uchwyt kolejki, do którego istnieje odwołanie, nie zapisuje kontekstu.

RC2098
(2098, X'832 ') Kontekst nie jest dostępny dla uchwytu kolejki, do którego się odwołuje.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2135
(2135, X'857 ') Niepoprawna struktura nagłówka dystrybucji.

RC2136
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

RC2137
(2137, X'859 ') Obiekt nie został pomyślnie otwarty.

RC2149
(2149, X'865 ') Niepoprawne struktury PCF.

RC2154
(2154, X'86A') Niepoprawna liczba rekordów.

RC2156
(2156, X'86C') Rekordy odpowiedzi są niepoprawne.

RC2158
(2158, X'86E') Niepoprawne flagi rekordu komunikatu.

RC2159
(2159, X'86F') Niepoprawne rekordy umieszczonych komunikatów.

RC2161
(2161, X'871 ') wygaszanie menedżera kolejek.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2173
(2173, X'87D') Niepoprawna struktura opcji umieszczania komunikatu.

RC2185
(2185, X'889 ') Niespójna specyfikacja trwałości.

RC2188
(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

RC2189
(2189, X'88D') Rozstrzyganie nazwy klastra nie powiodło się.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2241
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

RC2242
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

RC2245
(2245, X'8C5') Niespójna specyfikacja jednostki pracy.

RC2248
(2248, X'8C8') Niepoprawne rozszerzenie deskryptora komunikatu.

RC2249
(2249, X'8C9') Niepoprawne flagi komunikatu.

RC2250
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

RC2251
(2251, X'8CB') Przesunięcie segmentu komunikatu jest niepoprawne.

RC2252
(2252, X'8CC') Niepoprawna oryginalna długość.

RC2253
(2253, X'8CD') Długość danych w segmencie komunikatu wynosi zero.

RC2255
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

RC2257
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

RC2258
(2258, X'8D2') Niepoprawny identyfikator grupy.

RC2266
(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

RC2269
(2269, X'8DD') Błąd zasobu klastra.

RC2270
(2270, X'8DE') Brak dostępnych kolejek docelowych.

RC2420
(2420) Wywołanie MQPUT zostało wykonane, ale dane komunikatu zawierają niepoprawną strukturę MQEPH.

RC2479
(2479, X'9AF') Nie można zachować publikacji.

RC2480
(2480, X'9B0') Typ docelowy uległ zmianie: kolejka aliasowa odwołuje się do kolejki, ale teraz odwołuje się do tematu.

RC2502
(2502, X'9C6') Publikowanie nie powiodło się, a publikacja nie została dostarczona do żadnego z subskrybentów

RC2551
(2551, X'9F7') Określony łańcuch wyboru jest niedostępny.

RC2554
(2554, X'9FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat powinien zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQPUT(HCONN : HOBJ : MSGDSC : PMO :
C                                     BUFLEN : BUFFER : CMPCOD :
C                                     REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQPUT            PR                  EXTPROC('MQPUT')
D* Connection handle
D HCONN                          10I 0 VALUE
D* Object handle
D HOBJ                           10I 0 VALUE
D* Message descriptor
D MSGDSC                        364A
D* Options that control the action of MQPUT
D PMO                           200A
D* Length of the message in Buffer
D BUFLEN                         10I 0 VALUE
D* Message data
D BUFFER                           *   VALUE
D* Completion code
D CMPCOD                         10I 0
D* Reason code qualifying CMPCOD
D REASON                         10I 0
```

IBM i MQPUT1 (umieść jeden komunikat) w IBM i

Wywołanie MQPUT1 umieszcza jeden komunikat w kolejce lub na liście dystrybucyjnej albo w temacie. Kolejka, lista dystrybucyjna lub temat nie muszą być otwarte.

Składnia

MQPUT1 *(HCONN, OBJDSC, MSGDSC, PMO, BUFLEN, BUFFER, CMPCOD, REASON)*

Użycie notatek

1. Zarówno wywołania MQPUT, jak i MQPUT1 mogą być używane do umieszczania komunikatów w kolejce. Wywołanie, które ma być używane, zależy od okoliczności:

- Wywołania MQPUT należy używać, gdy wiele komunikatów ma być umieszczonych w *tej samej* kolejce.

  Najpierw wysyłane jest wywołanie MQOPEN określające opcję OOOUT, a następnie jedno lub więcej żądań MQPUT w celu dodania komunikatów do kolejki; na końcu kolejka jest zamykana za pomocą wywołania MQCLOSE. Daje to lepszą wydajność niż wielokrotne użycie wywołania MQPUT1 .
- Wywołania MQPUT1 należy używać, gdy tylko *jeden* komunikat ma zostać umieszczony w kolejce.

  To wywołanie hermetyzuje wywołania MQOPEN, MQPUT i MQCLOSE w jedno wywołanie, minimalizując liczbę wywołań, które muszą zostać wykonane.

2. Jeśli aplikacja umieszcza sekwencję komunikatów w tej samej kolejce bez użycia grup komunikatów, Kolejność tych komunikatów jest zachowywana, jeśli spełnione są pewne warunki. Jednak w większości środowisk wywołanie MQPUT1 nie spełnia tych warunków i nie zachowuje kolejności komunikatów. W tych środowiskach należy użyć wywołania MQPUT. Szczegółowe informacje można znaleźć w uwagach dotyczących składni w opisie wywołania MQPUT.
3. Do umieszczania komunikatów na listach dystrybucyjnych można użyć wywołania MQPUT1 . Ogólne informacje na ten temat zawierają uwagi dotyczące używania wywołań MQOPEN i MQPUT.

   Podczas używania wywołania MQPUT1 występują następujące różnice:

   a. Jeśli rekordy odpowiedzi MQRR są udostępniane przez aplikację, muszą być udostępniane przy użyciu struktury MQOD; nie mogą być udostępniane przy użyciu struktury MQPMO.

   b. Kod przyczyny RC2137 nie jest nigdy zwracany przez funkcję MQPUT1 w rekordach odpowiedzi. Jeśli otwarcie kolejki nie powiedzie się, rekord odpowiedzi dla tej kolejki zawiera rzeczywisty kod przyczyny wynikający z operacji otwarcia.

      Jeśli operacja otwarcia kolejki zakończy się kodem zakończenia CCWARN, kod zakończenia i kod przyczyny w rekordzie odpowiedzi dla tej kolejki zostaną zastąpione przez kody zakończenia i przyczyny wynikające z operacji umieszczania.

      Podobnie jak w przypadku wywołań MQOPEN i MQPUT, menedżer kolejek ustawia rekordy odpowiedzi (jeśli zostały podane) tylko wtedy, gdy wynik wywołania nie jest taki sam dla wszystkich kolejek na liście dystrybucyjnej. Jest to wskazywane przez zakończenie wywołania z kodem przyczyny RC2136.
4. Jeśli do umieszczenia komunikatu w kolejce klastra używane jest wywołanie MQPUT1 , wywołanie zachowuje się tak, jakby w wywołaniu MQOPEN określono parametr OOBNDN.
5. Jeśli komunikat jest umieszczany z jedną lub większą liczbą struktur nagłówka IBM MQ na początku danych komunikatu aplikacji, menedżer kolejek wykonuje pewne sprawdzenia struktur nagłówka, aby sprawdzić, czy są one poprawne. Więcej informacji na ten temat zawierają uwagi dotyczące użycia wywołania MQPUT.
6. Jeśli wystąpi więcej niż jedna sytuacja ostrzegawcza (patrz parametr **CMPCOD**), zwracany jest *pierwszy* kod przyczyny z poniższej listy, który ma zastosowanie:

   a. RC2136

   b. RC2242

   c. RC2241

   d. RC2049 lub RC2104
7. Parametr **BUFFER** przedstawiony w przykładzie programowania w języku RPG jest zadeklarowany jako łańcuch, co ogranicza maksymalną długość parametru do 256 bajtów. Jeśli wymagany jest większy bufor, parametr powinien być zadeklarowany jako struktura lub jako pole w zbiorze fizycznym. Spowoduje to zwiększenie maksymalnej możliwej długości do około 32 kB.

Parametry

Wywołanie MQPUT1 ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

OBJDSC (MQOD)-wejście/wyjście

Deskryptor obiektu.

Jest to struktura identyfikująca kolejkę, do której dodawany jest komunikat. Szczegółowe informacje można znaleźć w sekcji “MQOD (deskryptor obiektu) w systemie IBM i” na stronie 1193.

Użytkownik musi mieć uprawnienia do otwierania kolejki dla danych wyjściowych. Kolejka **nie** może być kolejką modelową.

MSGDSC (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Ta struktura opisuje atrybuty wysyłanego komunikatu i odbiera informacje zwrotne po zakończeniu żądania umieszczenia. Szczegółowe informacje można znaleźć w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.

Jeśli aplikacja udostępnia strukturę MQMD version-1 , dane komunikatu mogą być poprzedzone strukturą MQMDE w celu określenia wartości pól, które istnieją w strukturze MQMD version-2 , ale nie w strukturze version-1. Pole *MDFMT* w strukturze MQMD musi być ustawione na wartość FMMDE, aby wskazać, że istnieje struktura MQMDE. Więcej informacji na temat zawiera sekcja “MQMDE (Rozszerzenie deskryptora komunikatu) w systemie IBM i” na stronie 1187.

PMO (MQPMO)-wejście/wyjście

Opcje sterujące działaniem komendy MQPUT1.

Szczegółowe informacje można znaleźć w sekcji “MQPMO (opcje umieszczania komunikatów) w systemie IBM i” na stronie 1208.

BUFLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość komunikatu w pliku *BUFFER*.

Zero jest poprawne i wskazuje, że komunikat nie zawiera danych aplikacji. Górny limit zależy od różnych czynników. Szczegółowe informacje można znaleźć w opisie parametru **BUFLEN** wywołania MQPUT.

BUFFER (1-bajtowy łańcuch bitowy x BUFLEN)-wejście

Dane komunikatu.

Jest to bufor zawierający dane komunikatu aplikacji, które mają zostać wysłane. Bufor powinien być wyrównany do granicy odpowiedniej dla rodzaju danych w komunikacie. Wyrównanie 4-bajtowe powinno być odpowiednie dla większości komunikatów (w tym komunikatów zawierających struktury nagłówków IBM MQ), ale niektóre komunikaty mogą wymagać bardziej rygorystycznego wyrównania. Na przykład komunikat zawierający 64-bitową binarną liczbę całkowitą może wymagać wyrównania 8-bajtowego.

Jeśli plik *BUFFER* zawiera dane znakowe, dane liczbowe lub oba te elementy, pola *MDCSI* i *MDENC* w parametrze **MSGDSC** powinny być ustawione na wartości odpowiednie dla danych; umożliwi to odbiorcy komunikatu konwersję danych (w razie potrzeby) na zestaw znaków i kodowanie używane przez odbiornik.

Uwaga: Wszystkie pozostałe parametry wywołania MQPUT1 muszą znajdować się w zestawie znaków określonym przez atrybut menedżera kolejek **CodedCharSetId** i kodowanie lokalnego menedżera kolejek określone przez ENNAT.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* ma wartość CCWARN:

RC2104
(2104, X'838 ') Opcja raportu w deskryptorze komunikatu nie została rozpoznana.

RC2136
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

RC2049
(2049, X'801 ') Priorytet komunikatu przekracza maksymalną obsługiwaną wartość.

RC2241
(2241, X'8C1') Grupa komunikatów nie jest kompletna.

RC2242
(2242, X'8C2') Komunikat logiczny nie jest kompletny.

Jeśli *CMPCOD* to CCFAIL:

RC2001
(2001, X'7D1') Kolejka podstawowa aliasu nie jest poprawnym typem.

RC2004
(2004, X'7D4') Niepoprawny parametr buforu.

RC2005
(2005, X'7D5') Niepoprawny parametr długości buforu.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2013
(2013, X'7DD') Niepoprawny czas ważności.

RC2014
(2014, X'7DE') Kod zapisu czynności jest niepoprawny.

RC2017
(2017, X'7E1') Brak dostępnych uchwytów.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2024
(2024, X'7E8') W bieżącej jednostce pracy nie można obsłużyć więcej komunikatów.

RC2026
(2026, X'7EA') Niepoprawny deskryptor komunikatu.

RC2027
(2027, X'7EB') Brak kolejki odpowiedzi.

RC2029
(2029, X'7ED') Niepoprawny typ komunikatu w deskryptorze komunikatu.

RC2030
(2030, X'7EE') Długość komunikatu przekracza maksimum dla kolejki.

RC2031
(2031, X'7EF') Długość komunikatu jest większa niż wartość maksymalna dla menedżera kolejek.

RC2035
(2035, X'7F3') Brak uprawnień dostępu.

RC2042
(2042, X'7FA') Obiekt jest już otwarty z opcjami powodującymi konflikt.

RC2043
(2043, X'7FB') Niepoprawny typ obiektu.

RC2044
(2044, X'7FC') Niepoprawna struktura deskryptora obiektu.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2047
(2047, X'7FF') Trwałość jest niepoprawna.

RC2048
(2048, X'800 ') Kolejka nie obsługuje komunikatów trwałych.

RC2050
(2050, X'802 ') Priorytet komunikatu jest niepoprawny.

RC2051
(2051, X'803 ') Wywołania umieszczania zablokowane dla kolejki.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2053
(2053, X'805 ') Kolejka zawiera już maksymalną liczbę komunikatów.

RC2056
(2056, X'808 ') Brak miejsca na dysku dla kolejki.

RC2057
(2057, X'809 ') Niepoprawny typ kolejki.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2061
(2061, X'80D') Opcje raportu w deskryptorze komunikatu są niepoprawne.

RC2063
(2063, X'80F') Wystąpił błąd bezpieczeństwa.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2072
(2072, X'818 ') Obsługa punktu synchronizacji nie jest dostępna.

RC2082
(2082, X'822 ') Nieznana kolejka podstawowa aliasów.

RC2085
(2085, X'825 ') Nieznana nazwa obiektu.

RC2086
(2086, X'826 ') Nieznany menedżer kolejek obiektów.

RC2087
(2087, X'827 ') Nieznany zdalny menedżer kolejek.

RC2091
(2091, X'82B') Kolejka transmisji nie jest lokalna.

RC2092
(2092, X'82C') Kolejka transmisji z niewłaściwym użyciem.

RC2097
(2097, X'831 ') Uchwyt kolejki, do którego istnieje odwołanie, nie zapisuje kontekstu.

RC2098
(2098, X'832 ') Kontekst nie jest dostępny dla uchwytu kolejki, do którego się odwołuje.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2135
(2135, X'857 ') Niepoprawna struktura nagłówka dystrybucji.

RC2136
(2136, X'858 ') Zwrócono wiele kodów przyczyny.

RC2149
(2149, X'865 ') Niepoprawne struktury PCF.

RC2154
(2154, X'86A') Niepoprawna liczba rekordów.

RC2155
(2155, X'86B') Rekordy obiektów są niepoprawne.

RC2156
(2156, X'86C') Rekordy odpowiedzi są niepoprawne.

RC2158
(2158, X'86E') Niepoprawne flagi rekordu komunikatu.

RC2159
(2159, X'86F') Niepoprawne rekordy umieszczonych komunikatów.

RC2161
(2161, X'871 ') wygaszanie menedżera kolejek.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2173
(2173, X'87D') Niepoprawna struktura opcji umieszczania komunikatu.

RC2184
(2184, X'888 ') Niepoprawna nazwa kolejki zdalnej.

RC2188
(2188, X'88C') Wywołanie odrzucone przez wyjście obciążenia klastra.

RC2189
(2189, X'88D') Rozstrzyganie nazwy klastra nie powiodło się.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2196
(2196, X'894 ') Nieznana kolejka transmisji.

RC2197
(2197, X'895 ') Nieznana domyślna kolejka transmisji.

RC2198
(2198, X'896 ') Domyślna kolejka transmisji nie jest lokalna.

RC2199
(2199, X'897 ') Błąd użycia domyślnej kolejki transmisji.

RC2258
(2258, X'8D2') Niepoprawny identyfikator grupy.

RC2248
(2248, X'8C8') Niepoprawne rozszerzenie deskryptora komunikatu.

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2249
(2249, X'8C9') Niepoprawne flagi komunikatu.

RC2250
(2250, X'8CA') Numer kolejny komunikatu jest niepoprawny.

RC2251
(2251, X'8CB') Przesunięcie segmentu komunikatu jest niepoprawne.

RC2252
(2252, X'8CC') Niepoprawna oryginalna długość.

RC2253
(2253, X'8CD') Długość danych w segmencie komunikatu wynosi zero.

RC2255
(2255, X'8CF') Jednostka pracy niedostępna dla menedżera kolejek.

RC2257
(2257, X'8D1') Podano niewłaściwą wersję MQMD.

RC2266
(2266, X'8DA') Wyjście obciążenia klastra nie powiodło się.

RC2269
(2269, X'8DD') Błąd zasobu klastra.

RC2270
(2270, X'8DE') Brak dostępnych kolejek docelowych.

RC2420
(2420) Wywołanie MQPUT1 zostało wykonane, ale dane komunikatu zawierają niepoprawną strukturę MQEPH.

RC2551
(2551, X'9F7') Określony łańcuch wyboru jest niedostępny.

RC2554
(2554, X'9FA') Nie można przeanalizować treści komunikatu w celu określenia, czy komunikat powinien zostać dostarczony do subskrybenta z rozszerzonym selektorem komunikatów.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQPUT1(HCONN : OBJDSC : MSGDSC :
C                                      PMO : BUFLEN : BUFFER :
C                                      CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQPUT1           PR                  EXTPROC('MQPUT1')
D* Connection handle
D HCONN                        10I 0 VALUE
D* Object descriptor
D OBJDSC                      468A
D* Message descriptor
D MSGDSC                      364A
D* Options that control the action of MQPUT1
D PMO                         200A
```

```
     D* Length of the message in BUFFER
     D BUFLEN                        10I 0 VALUE
     D* Message data
     D BUFFER                          *   VALUE
     D* Completion code
     D CMPCOD                        10I 0
     D* Reason code qualifying CMPCOD
     D REASON                        10I 0
```

IBM i MQSET (ustawienie atrybutów obiektu) w systemie IBM i

Wywołanie MQSET służy do zmiany atrybutów obiektu reprezentowanego przez uchwyt. Obiekt musi być kolejką.

Składnia

MQSET *(HCONN, HOBJ, SELCNT, SELS, IACNT, INTATR, CALEN, CHRATR, CMPCOD, REASON)*

Użycie notatek

1. Za pomocą tego wywołania aplikacja może określić tablicę atrybutów całkowitych, kolekcję łańcuchów atrybutów znakowych lub jedno i drugie. Jeśli nie wystąpią żadne błędy, wszystkie podane atrybuty są ustawiane równocześnie. Jeśli wystąpi błąd (na przykład, jeśli selektor jest niepoprawny lub podjęto próbę ustawienia atrybutu na niepoprawną wartość), wywołanie nie powiedzie się i nie zostaną ustawione żadne atrybuty.
2. Wartości atrybutów można określić przy użyciu wywołania MQINQ. Szczegółowe informacje na ten temat zawiera sekcja "MQINQ (zapytanie o atrybuty obiektu) w systemie IBM i" na stronie 1345 .

   Uwaga: Nie wszystkie atrybuty z wartościami, do których można uzyskać dostęp za pomocą wywołania MQINQ, mogą mieć zmienione wartości za pomocą wywołania MQSET. Na przykład za pomocą tego wywołania nie można ustawić żadnych atrybutów obiektu procesu ani menedżera kolejek.
3. Zmiany atrybutów są zachowywane po restarcie menedżera kolejek (inne niż zmiany tymczasowych kolejek dynamicznych, które nie są zachowywane po restarcie menedżera kolejek).
4. Nie można zmieniać atrybutów kolejki modelowej za pomocą wywołania MQSET. Jeśli jednak kolejka modelowa jest otwierana za pomocą wywołania MQOPEN z opcją MQOO_SET, można użyć wywołania MQSET do ustawienia atrybutów dynamicznej kolejki lokalnej utworzonej przez wywołanie MQOPEN.
5. Jeśli ustawiany obiekt jest kolejką klastra, musi istnieć lokalna instancja kolejki klastra, aby otwarcie powiodło się.

Więcej informacji na temat atrybutów obiektu zawiera sekcja:

Parametry

Wywołanie MQSET ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość HCONN została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt obiektu.

Ten uchwyt reprezentuje obiekt kolejki z atrybutami, które mają zostać ustawione. Uchwyt został zwrócony przez poprzednie wywołanie MQOPEN, które określało opcję OOSET.

SELCNT (10-cyfrowa liczba całkowita ze znakiem)-wejście

Liczba selektorów.

Jest to liczba selektorów, które są dostarczane w tablicy SELS . Jest to liczba atrybutów, które mają zostać ustawione. Zero jest poprawną wartością. Maksymalna dozwolona liczba to 256.

SELS (10-cyfrowa liczba całkowita ze znakiem x SELCNT)-wejście

Tablica selektorów atrybutów.

Jest to tablica selektorów atrybutów **SELCNT** . Każdy selektor identyfikuje atrybut (liczbę całkowitą lub znak) z wartością, która ma zostać ustawiona.

Każdy selektor musi być poprawny dla typu kolejki reprezentowanego przez HOBJ . Dozwolone są tylko niektóre wartości IA* i CA*; wartości te są wymienione w dalszej części tej sekcji.

Selektory można określić w dowolnej kolejności. Wartości atrybutów, które odpowiadają selektorom atrybutów będącym liczbami całkowitymi (IA* selectors), muszą być określone w produkcie INTATR w tej samej kolejności, w jakiej występują one w produkcie SELS. Wartości atrybutów odpowiadające selektorom atrybutów znakowych (CA* selectors) muszą być określone w pliku CHRATR w tej samej kolejności, w jakiej występują te selektory. Selektory IA* mogą być przeplatane z selektorami CA*; ważna jest tylko kolejność względna w obrębie każdego typu.

Określenie tego samego selektora więcej niż raz nie jest błędem. W takim przypadku zostanie zastosowana ostatnia wartość określona dla konkretnego selektora.

Uwaga:

1. Selektory atrybutów całkowitych i znakowych są przydzielane w dwóch różnych zakresach; selektory IA* rezydują w zakresie od IAFRST do IALAST, a selektory CA* w zakresie od CAFRST do CALAST.

   Dla każdego zakresu stałe IALSTU i CALSTU definiują najwyższą wartość, którą zaakceptuje menedżer kolejek.
2. Jeśli najpierw wystąpią wszystkie selektory IA*, można użyć tych samych numerów elementów do adresowania odpowiednich elementów w macierzach SELS i INTATR .

Atrybuty, które można ustawić, są wymienione w poniższej tabeli. Przy użyciu tego wywołania nie można ustawić żadnych innych atrybutów. W przypadku selektorów atrybutów CA* w nawiasach podano stałą definiującą długość łańcucha (w bajtach), który jest wymagany w elemencie CHRATR .

Tabela 751. Selektory atrybutów MQSET dla kolejek

Selektor	Opis	Uwaga
KATRGD	Dane wyzwalacza (LNTRGD).	“2” na stronie 1393
IADIST	Obsługa listy dystrybucyjnej.	“1” na stronie 1393
IAIGET	Określa, czy operacje pobierania są dozwolone.	

Tabela 751. Selektory atrybutów MQSET dla kolejek (kontynuacja)		
Selektor	**Opis**	**Uwaga**
IAIPUT (wprowadzanie)	Określa, czy operacje umieszczania (put) są dozwolone.	
IATRGC	Element sterujący wyzwalacza.	“2” na stronie 1393
IATRGD	Wyzwalacz uruchamiany zapełnieniem.	“2” na stronie 1393
IATRGP	Priorytet komunikatu progowego dla wyzwalaczy.	“2” na stronie 1393
IATRGT	Typ wyzwalacza.	“2” na stronie 1393

Uwagi:

1. Obsługiwane tylko na następujących platformach:
   - AIX
   - IBM i
   - Windows

   i dla klientów IBM MQ połączonych z tymi systemami.
2. Nieobsługiwane w systemie VSE/ESA.

IACNT (10-cyfrowa liczba całkowita ze znakiem)-wejście

Liczba atrybutów całkowitych.

Jest to liczba elementów w tablicy INTATR i musi to być co najmniej liczba selektorów IA* w parametrze **SELS** . Zero jest poprawną wartością, jeśli nie ma żadnej wartości.

INTATR (10-digit signed integ x rxIACNT)-wejście

Tablica atrybutów całkowitych.

Jest to tablica wartości atrybutów całkowitoliczbowych IACNT . Wartości tych atrybutów muszą być w tej samej kolejności, co selektory IA* w tablicy SELS .

CALEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość buforu atrybutów znakowych.

Jest to długość w bajtach parametru **CHRATR** , która musi być sumą długości atrybutów znakowych określonych w tablicy SELS . Zero jest poprawną wartością, jeśli w pliku SELSnie ma selektorów CA*.

CHRATR (1-bajtowy łańcuch znaków x CALEN)-wejście

Atrybuty znakowe.

Jest to bufor zawierający połączone ze sobą wartości atrybutów znakowych. Długość buforu jest określona przez parametr **CALEN** .

Atrybuty znaków muszą być podane w tej samej kolejności, co selektory CA* w tablicy SELS . Długość każdego atrybutu znakowego jest stała (patrz sekcja SELS). Jeśli wartość, która ma być ustawiona dla atrybutu, zawiera mniej niepustych znaków niż zdefiniowana długość atrybutu, wartość podana w polu CHRATR musi być dopełniona po prawej stronie odstępami, aby wartość atrybutu była zgodna ze zdefiniowaną długością atrybutu.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający CMPCOD.

Jeśli CMPCOD to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli CMPCOD to CCFAIL:

RC2219
(2219, X'8AB') Wywołanie MQI zostało ponownie wprowadzone przed zakończeniem poprzedniego wywołania.

RC2006
(2006, X'7D6') Niepoprawna długość atrybutów znakowych.

RC2007
(2007, X'7D7') Niepoprawny łańcuch atrybutów znaku.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2019
(2019, X'7E3') Uchwyt obiektu jest niepoprawny.

RC2020
(2020, X'7E4') Niepoprawna wartość atrybutu kolejki zablokowanej-get lub zablokowanej-put.

RC2021
(2021, X'7E5') Niepoprawna liczba atrybutów liczby całkowitej.

RC2023
(2023, X'7E7') Niepoprawna tablica atrybutów całkowitych.

RC2040
(2040, X'7F8') Kolejka nie jest otwarta do ustawiania.

RC2041
(2041, X'7F9') Definicja obiektu została zmieniona od momentu otwarcia.

RC2101
(2101, X'835 ') Obiekt uszkodzony.

RC2052
(2052, X'804 ') Kolejka została usunięta.

RC2058
(2058, X'80A') Nazwa menedżera kolejek jest niepoprawna lub nieznana.

RC2059
(2059, X'80B') Menedżer kolejek nie jest dostępny dla połączenia.

RC2162
(2162, X'872 ') Menedżer kolejek jest zamykany.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2065
(2065, X'811 ') Niepoprawna liczba selektorów.

RC2067
(2067, X'813 ') Niepoprawny selektor atrybutu.

RC2066
(2066, X'812 ') Zbyt duża liczba selektorów.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2075
(2075, X'81B') Niepoprawna wartość atrybutu trigger-control.

RC2076
(2076, X'81C') Niepoprawna wartość atrybutu trigger-depth.

RC2077
(2077, X'81D') Niepoprawna wartość atrybutu trigger-message-priority.

RC2078
(2078, X'81E') Niepoprawna wartość atrybutu typu wyzwalacza.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSET(HCONN : HOBJ : SELCNT :
C                                     SELS(1) : IACNT : INTATR(1) :
C                                     CALEN : CHRATR : CMPCOD :
C                                     REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQSET            PR                  EXTPROC('MQSET')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Object handle
D HOBJ                          10I 0 VALUE
D* Count of selectors
D SELCNT                        10I 0 VALUE
D* Array of attribute selectors
D SELS                          10I 0
D* Count of integer attributes
D IACNT                         10I 0 VALUE
D* Array of integer attributes
D INTATR                        10I 0
D* Length of character attributes buffer
D CALEN                         10I 0 VALUE
D* Character attributes
D CHRATR                          *   VALUE
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQSETMP (ustawienie właściwości uchwytu komunikatu) w systemie IBM i

Wywołanie MQSETMP ustawia lub modyfikuje właściwość uchwytu komunikatu.

Składnia

Komenda MQSETMP *(Hconn, Hmsg, SetPropOpts, Name, PropDesc, Type, ValueLength, Value, CompCode, Reason)*

Użycie notatek

- Tego wywołania można użyć tylko wtedy, gdy sam menedżer kolejek koordynuje jednostkę pracy. Może to być:
  - Lokalna jednostka pracy, w której zmiany mają wpływ tylko na zasoby IBM MQ .
  - Globalna jednostka pracy, w której zmiany mogą mieć wpływ na zasoby należące do innych menedżerów zasobów, a także na zasoby IBM MQ .

  Więcej informacji na temat lokalnych i globalnych jednostek pracy zawiera sekcja "MQBEGIN (rozpoczęcie jednostki pracy) w systemie IBM i" na stronie 1293.
- W środowiskach, w których menedżer kolejek nie koordynuje jednostki pracy, należy użyć odpowiedniego wywołania wycofania zamiast wywołania MQBACK. Środowisko może również obsługiwać niejawne wycofanie spowodowane nieprawidłowym zakończeniem działania aplikacji.
  - W systemie z/OSnależy użyć następujących wywołań:
    - Programy wsadowe (w tym programy IMS w języku DL/I) mogą używać wywołania MQBACK, jeśli jednostka pracy ma wpływ tylko na zasoby IBM MQ . Jeśli jednak jednostka pracy ma wpływ zarówno na zasoby IBM MQ , jak i zasoby należące do innych menedżerów zasobów (na przykład Db2), należy użyć wywołania SRRBACK udostępnianego przez usługę RRS (z/OS Recoverable Resource Service). Wywołanie SRRBACK wycofuje zmiany w zasobach należących do menedżerów zasobów, dla których włączono koordynację RRS.
    - Aplikacje CICS muszą używać komendy `EXEC CICS SYNCPOINT ROLLBACK` , aby wycofać jednostkę pracy. Nie należy używać wywołania MQBACK dla aplikacji CICS .
    - Aplikacje IMS (inne niż programy wsadowe DL/I) muszą używać wywołań języka IMS , takich jak `ROLB` , do tworzenia kopii zapasowej jednostki pracy. Nie należy używać wywołania MQBACK dla aplikacji IMS (innych niż programy wsadowe DL/I).
  - W systemie IBM itego wywołania należy używać dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek. Oznacza to, że definicja kontroli transakcji nie może istnieć na poziomie zadania, co oznacza, że komenda STRCMTCTL z parametrem **CMTSCOPE(*JOB)** nie została wydana dla zadania.
- Jeśli aplikacja kończy się z niezatwierdzonymi zmianami w jednostce pracy, rozdysponowanie tych zmian zależy od tego, czy aplikacja kończy się normalnie, czy nieprawidłowo. Więcej informacji na ten temat zawierają uwagi dotyczące składni w sekcji "MQDISC (Rozłączenie menedżera kolejek) w systemie IBM i" na stronie 1330 .
- Gdy aplikacja umieszcza lub pobiera komunikaty w grupach lub segmentach komunikatów logicznych, menedżer kolejek zachowuje informacje dotyczące grupy komunikatów i komunikatu logicznego dla ostatnich pomyślnych wywołań MQPUT i MQGET. Te informacje są powiązane z uchwytem kolejki i obejmują takie elementy, jak:
  - Wartości pól *GroupId*, *MsgSeqNumber*, *Offset*i *MsgFlags* w strukturze MQMD.

– Określa, czy komunikat jest częścią jednostki pracy.
– Dla wywołania MQPUT: określa, czy komunikat jest trwały, czy nietrwały.

Menedżer kolejek przechowuje trzy zestawy informacji o grupach i segmentach, po jednym zestawie dla każdego z następujących elementów:

– Ostatnie pomyślne wywołanie MQPUT (może być częścią jednostki pracy).
– Ostatnie pomyślne wywołanie MQGET, które usunęło komunikat z kolejki (może to być część jednostki pracy).
– Ostatnie pomyślne wywołanie MQGET, które przeglądnięło komunikat w kolejce (nie może być częścią jednostki pracy).

Jeśli aplikacja umieszcza lub pobiera komunikaty jako część jednostki pracy, a następnie decyduje się wycofać jednostkę pracy, informacje o grupie i segmencie są przywracane do wartości, która była wcześniej używana:

– Informacje powiązane z wywołaniem MQPUT zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQPUT dla tego uchwytu kolejki w bieżącej jednostce pracy.
– Informacje powiązane z wywołaniem MQGET zostaną przywrócone do wartości sprzed pierwszego pomyślnego wywołania MQGET dla tego uchwytu kolejki w bieżącej jednostce pracy.

Kolejki, które zostały zaktualizowane przez aplikację po uruchomieniu jednostki pracy, ale poza zasięgiem jednostki pracy, nie mają odtworzonych informacji o grupach i segmentach, jeśli jednostka pracy została wycofana.

Przywrócenie poprzedniej wartości informacji o grupie i segmencie po wycofaniu jednostki pracy umożliwia aplikacji rozłożenie dużej grupy komunikatów lub dużego komunikatu logicznego składającego się z wielu segmentów na kilka jednostek pracy oraz zrestartowanie w odpowiednim punkcie grupy komunikatów lub komunikatu logicznego, jeśli jedna z jednostek pracy nie powiedzie się.

Użycie kilku jednostek pracy może być korzystne, jeśli lokalny menedżer kolejek ma tylko ograniczoną pamięć kolejki. Jednak aplikacja musi zachować wystarczającą ilość informacji, aby w przypadku awarii systemu możliwe było zrestartowanie umieszczania lub pobierania komunikatów w odpowiednim miejscu.

Szczegółowe informacje na temat restartowania w poprawnym punkcie po awarii systemu zawiera opis opcji PMLOGO w sekcji PMOPT (10-cyfrowa liczba całkowita ze znakiem)oraz opis opcji GMLOGO w sekcji GMOPT (10-cyfrowa liczba całkowita ze znakiem).

Pozostałe uwagi dotyczące użycia mają zastosowanie tylko wtedy, gdy menedżer kolejek koordynuje jednostki pracy:

- Jednostka pracy ma taki sam zasięg jak uchwyt połączenia. Wszystkie wywołania IBM MQ , które mają wpływ na konkretną jednostkę pracy, muszą być wykonywane przy użyciu tego samego uchwytu połączenia. Wywołania wykonane przy użyciu innego uchwytu połączenia (na przykład wywołania wykonane przez inną aplikację) mają wpływ na inną jednostkę pracy. Informacje na temat zasięgu uchwytów połączenia zawiera sekcja HCONN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe .
- To wywołanie ma wpływ tylko na komunikaty, które zostały umieszczone lub pobrane jako część bieżącej jednostki pracy.
- Aplikacja długotrwała, która wywołuje wywołania MQGET, MQPUT lub MQPUT1 w jednostce pracy, ale nigdy nie wywołuje wywołania zatwierdzenia lub wycofania, może zapełniać kolejki komunikatami, które nie są dostępne dla innych aplikacji. Aby zabezpieczyć się przed taką możliwością, administrator musi ustawić atrybut **MaxUncommittedMsgs** menedżera kolejek na wartość, która jest na tyle niska, aby zapobiec zapełnianiu kolejek przez aplikacje niedziałające, ale na tyle wysoka, aby umożliwić poprawne działanie oczekiwanych aplikacji przesyłania komunikatów.

Parametry

Wywołanie MQSETMP ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek.

Wartość musi być zgodna z uchwytem połączenia, który został użyty do utworzenia uchwytu komunikatu określonego w parametrze **HMSG** .

Jeśli uchwyt komunikatu został utworzony za pomocą HCUNAS, należy nawiązać poprawne połączenie w wątku, ustawiając właściwość uchwytu komunikatu. W przeciwnym razie wywołanie nie powiedzie się z kodem przyczyny RC2009 .

HMSG (20-cyfrowa liczba całkowita ze znakiem)-wejście

Jest to uchwyt komunikatu, który ma zostać zmodyfikowany. Wartość została zwrócona przez poprzednie wywołanie MQCRTMH.

SETOPT (MQSMPO)-wejście

Sterowanie sposobem ustawiania właściwości komunikatu.

Ta struktura umożliwia aplikacjom określanie opcji sterujących ustawianiem właściwości komunikatu. Struktura jest parametrem wejściowym wywołania MQSETMP. Więcej informacji na ten temat zawiera sekcja MQSMPO .

PRNAME (MQCHARV)-wejście

Jest to nazwa właściwości, która ma zostać ustawiona.

Więcej informacji na temat używania nazw właściwości zawiera sekcja Nazwy właściwości i sekcja Ograniczenia dotyczące nazw właściwości .

PRPDSC (MQPD)-wejście/wyjście

Ta struktura jest używana do definiowania atrybutów właściwości, w tym:

- co się stanie, jeśli właściwość nie jest obsługiwana
- kontekst komunikatu, do którego należy właściwość
- Komunikaty, do których właściwość jest kopiowana podczas przepływu

Więcej informacji na temat tej struktury zawiera sekcja MQPD .

TYPE (10-cyfrowa liczba całkowita ze znakiem)-wejście

Typ danych ustawianej właściwości. Może to być jedna z następujących wartości:

TYPBOL
Wartość boolowska. *ValueLength* musi mieć wartość 4.

TYPBST
Łańcuch bajtowy. *ValueLength* musi mieć wartość zero lub większą.

TYPI8
8-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 1.

TYPI16
16-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 2.

TYPI32
32-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 4.

TYPI64
64-bitowa liczba całkowita ze znakiem. *ValueLength* musi mieć wartość 8.

TYPF32
32-bitowa liczba zmiennopozycyjna. *ValueLength* musi mieć wartość 4.

TYPF64
64-bitowa liczba zmiennopozycyjna. *ValueLength* musi mieć wartość 8.

TYPSTR
Łańcuch znaków. *ValueLength* musi mieć wartość zero lub większą albo wartość specjalną VLNULL.

TYPNUL
Właściwość istnieje, ale ma wartość null. *ValueLength* musi mieć wartość zero.

VALLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Długość (w bajtach) wartości właściwości w parametrze *wartość* .

Zero jest poprawne tylko dla wartości null lub dla łańcuchów lub łańcuchów bajtowych. Wartość zero wskazuje, że właściwość istnieje, ale wartość nie zawiera żadnych znaków ani bajtów.

Wartość musi być większa lub równa zero lub następująca wartość specjalna, jeśli parametr *Type* ma ustawioną wartość TYPSTR:

VLNULL
Wartość jest ograniczona przez pierwszą wartość null napotkaną w łańcuchu. Wartość NULL nie jest uwzględniana jako część łańcucha. Ta wartość jest niepoprawna, jeśli nie ustawiono również parametru TYPSTR.

Uwaga: Znak o kodzie zero używany do kończenia łańcucha, jeśli ustawiona jest wartość VLNULL, jest to znak o kodzie zero z zestawu znaków wartości.

VALUE (1-bajtowy łańcuch bitowy x VALLEN)-wejście

Wartość właściwości, która ma zostać ustawiona. Bufor musi być wyrównany do granicy odpowiedniej dla rodzaju danych w wartości.

W języku programowania C parametr jest zadeklarowany jako wskaźnik do unieważnienia; jako parametr można podać adres dowolnego typu danych.

Jeśli *ValueLength* ma wartość zero, *Wartość* nie jest przywoływana. W takim przypadku adres parametru przekazywany przez programy napisane w języku C lub w asemblerze System/390 może mieć wartość NULL.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający *CMPCOD*.

Jeśli opcja *CMPCOD* ma wartość CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli opcja *CMPCOD* ma wartość CCWARN:

RC2421
(2421, X'0975 ') Nie można przeanalizować folderu MQRFH2 zawierającego właściwości.

Jeśli *CMPCOD* to CCFAIL:

RC2204
(2204, X'089C') Adapter nie jest dostępny.

RC2130
(2130, X'852 ') Nie można załadować modułu usługi adaptera.

RC2157
(2157, X'86D') Główne i główne identyfikatory ASID różnią się.

RC2004
(2004, X'07D4') Niepoprawny parametr wartości.

RC2005
(2005, X'07D5') Niepoprawny parametr długości wartości.

RC2219
(2219, X'08AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2460
(2460, X'099C') Wskaźnik uchwytu komunikatu jest niepoprawny.

RC2499
(2499, X'09C3') Uchwyt komunikatu jest już używany.

RC2046
(2046, X'07FE') Opcje są niepoprawne lub niespójne.

RC2482
(2482, X'09B2') Niepoprawna struktura deskryptora właściwości.

RC2442
(2442, X'098A') Niepoprawna nazwa właściwości.

RC2473
(2473, X'09A9') Niepoprawny typ danych właściwości.

RC2472
(2472, X'09A8') Wystąpił błąd formatu liczb w danych wartości.

RC2463
(2463, X'099F') Niepoprawna struktura opcji ustawiania właściwości komunikatu.

RC2111
(2111, X'083F') Niepoprawny identyfikator kodowanego zestawu znaków nazwy właściwości.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat zawiera sekcja “Kody powrotu dla IBM i (ILE RPG)” na stronie 1471.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSETMP(HCONN : HMSG : SETOPT :
                                        PRNAME : PRPDSC :
                                        TYPE : VALLEN : VALUE :
                                        CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
DMQSETMP          PR                  EXTPROC('MQSETMP')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Message handle
D HMSG                          10I 0 VALUE
D* Options that control the action of MQSETMP
D SETOPT                        20A
D* Property name
D PRNAME                        32A
D* Property descriptor
D PRPDSC                        24A
D* Property data type
D TYPE                          10I 0 VALUE
D* Length of the Value area
D VALLEN                        10I 0 VALUE
D* Property value
D VALUE                           *   VALUE
D* Completion code
```

```
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQSTAT (Odtworzenie informacji o statusie) w systemie IBM i

Użyj wywołania MQSTAT, aby pobrać informacje o statusie. Typ zwracanych informacji o statusie jest określany przez wartość STYPE określoną w wywołaniu.

Składnia

(HCONN, STYPE, STAT, CMPCOD, REASON) MQSTAT

Użycie notatek

1. Wywołanie komendy MQSTAT określające typ STATAPT zwraca informacje o poprzednich asynchronicznych operacjach MQPUT i MQPUT1 . Struktura MQSTAT przekazana w wywołaniu jest zakończona z pierwszym zarejestrowanym asynchronicznym ostrzeżeniem lub informacją o błędzie dla tego połączenia. Jeśli kolejne błędy lub ostrzeżenia następują po pierwszym, zwykle nie zmieniają one tych wartości. Jeśli jednak wystąpi błąd z kodem zakończenia CCWARN, zostanie zwrócony kolejny błąd z kodem zakończenia CCFAIL.
2. Jeśli od momentu nawiązania połączenia lub od ostatniego wywołania MQSTAT nie wystąpiły żadne błędy, zwracana jest wartość CMPCOD równa CCOK i REASON of RCNONE.
3. Liczba wywołań asynchronicznych, które zostały przetworzone pod uchwytem połączenia, jest zwracana przy użyciu trzech liczników: STSPSC, STSPWC i STSPFC. Liczniki te są zwiększane przez menedżera kolejek za każdym razem, gdy operacja asynchroniczna jest przetwarzana pomyślnie, ma ostrzeżenie lub kończy się niepowodzeniem (należy pamiętać, że w celu rozliczania operacji umieszczania na liście dystrybucyjnej jest wykonywane raz dla kolejki docelowej, a nie raz dla listy dystrybucyjnej).
4. Pomyślne wywołanie komendy MQSTAT powoduje zresetowanie wszystkich poprzednich informacji o błędach lub liczników.

Parametry

Wywołanie MQSTAT ma następujące parametry:

Hconn (MQHCONN)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

STYPE (10-cyfrowa liczba całkowita ze znakiem)-wejście

Typ żądanych informacji o statusie. Jedyną poprawną wartością jest:

STATAPTa
Zwraca informacje o poprzednich asynchronicznych operacjach umieszczania.

STS (MQSTS)-wejście/wyjście

Struktura informacji o statusie. Szczegółowe informacje można znaleźć w sekcji "MQSTS (struktura raportowania statusu) w systemie IBM i" na stronie 1269.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Zakończono pomyślnie.

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2374
(2374, X' 946 ') Wyjście funkcji API nie powiodło się

RC2183
(2183, X'887 ') Nie można załadować wyjścia funkcji API.

RC2219
(2219, X'8AB') Wywołanie MQI wprowadzone przed zakończeniem poprzedniego wywołania.

RC2009
(2009, X'7D9') Połączenie z menedżerem kolejek zostało utracone.

RC2203
(2203, X'89B') Połączenie jest zamykane.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2162
(2162, X'872 ') Zatrzymywanie menedżera kolejek

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2430
(2430, X'97E') Błąd typu MQSTAT.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2424
(2424, X' 978 ') Błąd w strukturze MQSTS

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

RC2298
(2298, X'8FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

Szczegółowe informacje na temat tych kodów znajdują się w następujących sekcjach:

- Komunikaty i kody przyczyny

Deklaracja RPG

```
C*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
C                     CALLP     MQSTAT(HCONN : ETYPE : ERR :
C                                      CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
DMQSTAT           PR                  EXTPROC('MQSTAT')
```

```
     D* Connection handle
     D HCONN                         10I 0 VALUE
     D* Status information type
     D STYPE                         10I 0 VALUE
     D* Status information
     D STATUS                       296A
     D* Completion code
     D CMPCOD                        10I 0
     D* Reason code qualifying CompCode
     D REASON                        10I 0
```

IBM i MQSUB (rejestrowanie subskrypcji) w systemie IBM i

Wywołanie MQSUB rejestruje subskrypcję aplikacji w konkretnym temacie.

Składnia

(HCONN, SUBDSC, HOBJ, HSUB, CMPCOD, REASON) MQSUB

Użycie notatek

- Subskrypcja tematu jest tworzona przy użyciu nazwy skróconej predefiniowanego obiektu tematu, pełnej nazwy łańcucha tematu lub przez konkatenację dwóch części zgodnie z opisem w sekcji Łączenie łańcuchów tematów.
- Menedżer kolejek przeprowadza sprawdzanie zabezpieczeń w momencie wywołania MQSUB w celu sprawdzenia, czy identyfikator użytkownika, pod którym działa aplikacja, ma odpowiedni poziom uprawnień przed zezwoleniem na dostęp. Odpowiedni obiekt tematu jest znajdowany przy użyciu krótkiej nazwy podanej w wywołaniu lub przy użyciu najbliższej nazwy skróconej znalezionej w hierarchii tematów, jeśli podano długą nazwę. Sprawdzanie uprawnień do tego obiektu tematu jest wykonywane w celu upewnienia się, że uprawnienia do subskrybowania są ustawione oraz w kolejce docelowej w celu upewnienia się, że uprawnienia do danych wyjściowych są ustawione. Jeśli używana jest opcja SDMAN, oznacza to, że sprawdzanie uprawnień jest wykonywane w kolejce zarządzanej o nazwie powiązanej z tym obiektem tematu, a jeśli podano kolejkę niezarządzaną, oznacza to, że sprawdzanie uprawnień jest wykonywane w kolejce reprezentowanej przez parametr **HOBJ** .
- Zapytanie *HOBJ* zwracane w wywołaniu MQSUB, gdy używana jest opcja SOMAN, można uzyskać w celu określenia atrybutów, takich jak próg wycofania i nazwa nadmiernego ponownego wycofania. Można również zapytać o nazwę kolejki zarządzanej, ale nie należy próbować bezpośrednio otwierać tej kolejki.
- Subskrypcje mogą być grupowane, dzięki czemu tylko jedna publikacja może zostać dostarczona do grupy subskrypcji, nawet jeśli więcej niż jedna grupa jest zgodna z daną publikacją. Subskrypcje są grupowane przy użyciu opcji SOGRP i w celu grupowania subskrypcji muszą:
  - używać tej samej kolejki nazwanej (która nie jest używana z opcją SOMAN) w tym samym menedżerze kolejek-reprezentowanym przez parametr **HOBJ** w wywołaniu MQSUB
  - współużytkuj ten sam *SDCID*
  - być tego samego *SDSL*

  Te atrybuty definiują zestaw subskrypcji uważanych za należące do grupy, a także są atrybutami, których nie można zmienić, jeśli subskrypcja jest zgrupowana. Zmiana wartości parametru *SDSL* powoduje wyświetlenie wartości RC2512, a zmiana dowolnej innej wartości (którą można zmienić, jeśli subskrypcja nie jest zgrupowana) powoduje wyświetlenie wartości RC2515.
- Pola w strukturze MQSD są wypełniane po powrocie z wywołania MQSUB korzystającego z opcji SORES. Zwrócony kod MQSD można przekazać bezpośrednio do wywołania MQSUB, które korzysta z opcji SOALT z wszelkimi zmianami, które należy wprowadzić w subskrypcji zastosowanej do pliku MQSD. Niektóre pola mają specjalne uwagi, jak zaznaczono w tabeli.

Tabela 752. Dane wyjściowe MQSD z MQSUB	
Nazwa pola w MQSD	**Uwagi szczególne**
Opcje dostępu lub tworzenia	Żadna z tych opcji nie jest ustawiana podczas powrotu z wywołania MQSUB. W przypadku późniejszego ponownego wykorzystania usługi MQSD w wywołaniu usługi MQSUB wymagana opcja musi być jawnie ustawiona.
Opcje trwałości, opcje docelowe, opcje rejestracji i znaki wieloznaczne	Te opcje zostaną odpowiednio ustawione
Opcje publikacji	Te opcje zostaną odpowiednio ustawione, z wyjątkiem SONEWP, który ma zastosowanie tylko do SOCRE.
Inne opcje	Te opcje nie ulegają zmianie po powrocie z wywołania MQSUB. Sterują one sposobem wywoływania interfejsu API i nie są przechowywane z subskrypcją. Muszą one zostać ustawione zgodnie z wymaganiami w każdym kolejnym wywołaniu MQSUB z ponownym wykorzystaniem usługi MQSD.
ObjectName	To pole wejściowe pozostaje niezmienione w przypadku powrotu z wywołania MQSUB.
ObjectString	To pole wejściowe pozostaje niezmienione w przypadku powrotu z wywołania MQSUB. Jeśli zostanie podany bufor, w polu *SDRO* zwracana jest pełna nazwa tematu.
AlternateUserId i AlternateSecurityId	Te pola wejściowe są niezmienione w przypadku powrotu z wywołania MQSUB. Sterują one sposobem wywoływania interfejsu API i nie są przechowywane z subskrypcją. Muszą one zostać ustawione zgodnie z wymaganiami w każdym kolejnym wywołaniu MQSUB używającego ponownie usługi MQSD.
SubExpiry	Po powrocie z wywołania MQSUB przy użyciu opcji SORES to pole zostanie ustawione na pierwotny termin ważności subskrypcji, a nie na pozostały czas ważności. Jeśli usługa MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB przy użyciu opcji SOALT, utrata ważności subskrypcji zostanie zresetowana w celu ponownego rozpoczęcia zliczania.
SubName	To pole jest polem wejściowym wywołania MQSUB i nie jest zmieniane na wyjściu.
SubUserData i SelectionString	Te pola o zmiennej długości będą zwracane po wyjściu z wywołania MQSUB przy użyciu opcji SORES (jeśli podano bufor), a także przy użyciu dodatniej długości buforu w pliku *VCHRP*. Jeśli nie zostanie podany żaden bufor, w polu *VCHRL* tabeli MQCHARV.If podany bufor jest mniejszy niż obszar wymagany do zwrócenia pola, w podanym buforze zwracane są tylko *VCHRP* bajtów. Jeśli później usługa MQSD zostanie ponownie wykorzystana w wywołaniu MQSUB z użyciem opcji SOALT i nie zostanie podany bufor, ale zostanie podana niezerowa wartość *VCHRL* , jeśli ta długość będzie zgodna z istniejącą długością pola, nie zostanie wprowadzona żadna zmiana w tym polu.
SubCorrelId i znacznik PubAccounting	Jeśli identyfikator SOSCID nie jest używany, menedżer kolejek wygeneruje plik *SDCID* . Jeśli opcja SOSETI nie jest używana, program *SDACC* zostanie wygenerowany przez menedżer kolejek. Te pola zostaną zwrócone w danych MQSD z wywołania MQSUB przy użyciu opcji SORES. Jeśli są one generowane przez menedżer kolejek, wygenerowana wartość zostanie zwrócona w wywołaniu MQSUB przy użyciu opcji SOCRE lub SOALT.
PubPriority, SubLevel & PubApplIdentityData	Te pola zostaną zwrócone w danych MQSD.
Łańcuch ResObject	To pole wyjściowe zostanie zwrócone w danych MQSD tylko wtedy, gdy zostanie podany bufor.

Parametry

Wywołanie MQSUB ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

SUBDSC (MQSD)-wejście/wyjście

Jest to struktura identyfikująca obiekt z użyciem, który jest rejestrowany przez aplikację. Więcej informacji zawiera sekcja “MQSD (deskryptor subskrypcji) w systemie IBM i” na stronie 1249.

HOBJ (10-cyfrowa liczba całkowita ze znakiem)-wejście/wyjście

Ten uchwyt reprezentuje dostęp, który został ustanowiony w celu uzyskania komunikatów wysłanych do tej subskrypcji. Te komunikaty mogą być składowane w konkretnej kolejce lub menedżer kolejek może zostać poproszony o zarządzanie pamięcią masową bez konieczności tworzenia konkretnej kolejki.

Uchwyt obiektu.

Jeśli ma być używana konkretna kolejka, musi być powiązana z subskrypcją w czasie tworzenia. Można to zrobić na dwa sposoby:

- Przez podanie tego uchwytu podczas wywoływania MQSUB z opcją SDCRT. Jeśli ten uchwyt jest podany jako parametr wejściowy w wywołaniu, musi być poprawnym uchwytem obiektu zwróconym z poprzedniego wywołania MQOPEN kolejki przy użyciu co najmniej jednego z następujących parametrów: OOINP*, OOOUT (na przykład w przypadku kolejki zdalnej) lub opcji OOBRW. W przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2019. Nie może to być uchwyt obiektu do kolejki aliasowej, która jest tłumaczona na obiekt tematu. Jeśli tak, wywołanie kończy się niepowodzeniem z kodem powrotu RC2019
- Używając komendy MQSC DEFINE SUB i udostępniając tę komendę z nazwą obiektu kolejki.

Jeśli menedżer kolejek ma zarządzać przechowywaniem komunikatów wysyłanych do tej subskrypcji, należy to wskazać podczas tworzenia subskrypcji, używając opcji SOMAN i ustawiając wartość parametru HONONE. Menedżer kolejek zwraca uchwyt jako parametr wyjściowy w wywołaniu, a zwrócony uchwyt jest nazywany uchwytem zarządzanym. Jeśli określono parametr HONONE i nie określono również parametru SOMAN, wywołanie kończy się niepowodzeniem z kodem powrotu RC2019.

Uchwyt zarządzany, który jest zwracany przez menedżer kolejek, może być używany w wywołaniu MQGET lub MQCB, z opcjami przeglądania lub bez, w wywołaniu MQINQ lub w wywołaniu MQCLOSE. Nie można jej użyć w MQPUT, MQSET ani w kolejnej operacji MQSUB. Próba wykonania tej operacji kończy się niepowodzeniem z kodem powrotu RC2039 dla MQPUT, RC2040 dla MQSET lub RC2038 dla MQSUB.

Jeśli do wznowienia tej subskrypcji zostanie użyta opcja SORES w polu *OPTS* w strukturze MQSD, uchwyt może zostać zwrócony do aplikacji w tym parametrze, jeśli określono parametr HONONE. Tej opcji można użyć niezależnie od tego, czy subskrypcja używa zarządzanego uchwytu, czy nie. Może to być przydatne w przypadku subskrypcji utworzonych za pomocą komendy DEFINE SUB, jeśli uchwyt do kolejki subskrypcji ma być zdefiniowany w komendzie DEFINE SUB. Jeśli subskrypcja utworzona administracyjnie jest wznawiana, kolejka jest otwierana za pomocą OOINPQ i OOBRW. Jeśli potrzebne są inne opcje, aplikacja musi jawnie otworzyć kolejkę subskrypcji i udostępnić uchwyt obiektu w wywołaniu. Jeśli wystąpi problem z otwarciem kolejki, wywołanie nie powiedzie się i zostanie zgłoszony błąd RC2522. Jeśli podano parametr *HOBJ* , musi on być odpowiednikiem parametru *HOBJ* w oryginalnym wywołaniu MQSUB. Oznacza to, że jeśli udostępniono uchwyt obiektu zwrócony z wywołania MQOPEN, uchwyt musi być w tej samej kolejce, co poprzednio używany, w przeciwnym razie wywołanie nie powiedzie się i zostanie wyświetlony komunikat RC2019.

Jeśli ta subskrypcja jest zmieniana za pomocą opcji SOALT w polu *OPTS* w strukturze MQSD, można podać inną wartość *HOBJ* . Wszystkie publikacje, które zostały dostarczone do kolejki poprzednio określonej za pomocą tego parametru, pozostają w tej kolejce i aplikacja jest odpowiedzialna za pobranie tych komunikatów, jeśli parametr **HOBJ** reprezentuje teraz inną kolejkę.

Poniższa tabela zawiera podsumowanie użycia tego parametru z różnymi opcjami subskrypcji:

Tabela 753. Korzystanie z Hobj z różnymi opcjami subskrypcji		
Opcje	**Hobj**	**Opis**
SOCRT + SOMAN	Zignorowano na wejściu	Tworzy subskrypcję z pamięcią masową komunikatów zarządzaną przez menedżera kolejek.
SOCRT	Poprawny uchwyt obiektu	Tworzy subskrypcję udostępniającą konkretną kolejkę jako miejsce docelowe dla komunikatów.
SRES	HONONE.	Wznawia utworzoną wcześniej subskrypcję (zarządzaną lub nie) i zwraca uchwyt obiektu do użycia przez aplikację.
SRES	Poprawny, zgodny, uchwyt obiektu	Wznawia wcześniej utworzoną subskrypcję, która używa określonej kolejki jako miejsca docelowego dla komunikatów i używa uchwytu obiektu z konkretnymi opcjami otwarcia.
SOALT + SOMAN	HONONE.	Modyfikuje istniejącą subskrypcję, która wcześniej używała konkretnej kolejki, w celu zarządzania nią.
SOALT	Poprawny uchwyt obiektu	Zmienia istniejącą subskrypcję tak, aby używała konkretnej kolejki (z zarządzanej lub z innej konkretnej kolejki).

Niezależnie od tego, czy został on podany, czy zwrócony, w kolejnych wywołaniach MQGET należy określić parametr *HOBJ* , który ma odbierać publikacje.

Uchwyt *HOBJ* przestaje być poprawny, gdy zostanie dla niego wykonane wywołanie MQCLOSE lub gdy jednostka przetwarzania definiująca zasięg uchwytu zostanie zakończona. Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Więcej informacji na temat zasięgu uchwytu zawiera sekcja HCONN . Operacja MQCLOSE dla uchwytu *HOBJ* nie ma wpływu na uchwyt *HSUB* .

HSUB (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Ten uchwyt reprezentuje subskrypcję, która została wykonana. Może być używany do dwóch dodatkowych operacji:

- Można go użyć w kolejnym wywołaniu MQSUBRQ w celu wysłania publikacji, gdy podczas tworzenia subskrypcji używana jest opcja SOPUBR.
- Można go użyć w kolejnym wywołaniu MQCLOSE w celu usunięcia subskrypcji, która została wykonana. Uchwyt *HSUB* przestaje być poprawny w przypadku wywołania MQCLOSE lub zakończenia jednostki przetwarzania definiującej zasięg uchwytu. Zasięg zwróconego uchwytu obiektu jest taki sam jak zasięg uchwytu połączenia określonego w wywołaniu. Operacja MQCLOSE dla uchwytu *HSUB* nie ma wpływu na uchwyt *HOBJ* .

Tego uchwytu nie można przekazać do wywołania MQGET lub MQCB. Należy użyć parametru **HOBJ** . Przekazanie tego uchwytu do innych wyników wywołania IBM MQ w RC2019.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Pomyślne zakończenie

CWARN
Ostrzeżenie (częściowe zakończenie)

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CMPCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CMPCOD* to CCFAIL:

RC2019
(2019 X'07E3') Niepoprawny uchwyt obiektu

RC2046
(2046 X'07FE') Opcje niepoprawne lub niespójne

RC2085
(2085 X'0825 ') Nie można znaleźć zidentyfikowanego obiektu

RC2161
(2161 X'0871 ') wygaszanie menedżera kolejek

RC2298
(2298 X'08FA') Funkcja nie jest obsługiwana.

RC2424
(2424 X'0978 ') Niepoprawny deskryptor subskrypcji (MQSD)

RC2425
(2441 X' 979 ') Niepoprawny łańcuch tematu

RC2428
(2428 X'097C') Podana nazwa subskrypcji nie jest zgodna z istniejącymi subskrypcjami

RC2429
(2429 X'097D') Nazwa subskrypcji istnieje i jest używana przez inną aplikację

RC2431
(2431 X'097F') Pole danych SubUserjest niepoprawne

RC2432
(2432 X'0980 ') Subskrypcja istnieje

RC2434
(2434 X'0982 ') Nazwa subskrypcji jest zgodna z istniejącą subskrypcją

RC2440
(2440 X'0988 ') Pole SubName jest niepoprawne

RC2441
(2441 X'0989 ') Niepoprawne pole Objectstring

RC2435
(2435 X'0983 ') Atrybut nie może zostać zmieniony za pomocą komendy SDALT lub subskrypcja została utworzona za pomocą SDIMM.

RC2436
(2436 X'0984 ') Niepoprawna opcja SODUR

RC2459
(2459, X'99B') Błąd składniowy łańcucha wyboru.

RC2503
(2503 X'09C7') Wywołania MQSUB są obecnie zablokowane dla subskrybowanych tematów.

RC2519
(2519, X'9D7') Łańcuch wyboru nie jest zgodny z opisem sposobu użycia struktury MQCHARV.

RC2551
(2551, X'9F7') Określony łańcuch wyboru jest niedostępny.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSUB(HCONN : SUBDSC : HOBJ :
C                                     HSUB : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQSUB            PR                  EXTPROC('MQSUB')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Subscription descriptor
D SUBDSC                       400A
D* Object handle for queue
D HOBJ                          10I 0
D* Subscription object handle
D HSUB                          10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i MQSUBRQ (żądanie subskrypcji) w systemie IBM i

Wywołanie MQSUBRQ wysyła żądanie do subskrypcji.

- “Składnia” na stronie 1408
- “Użycie notatek” na stronie 1408
- “Parametry” na stronie 1409
- “Deklaracja RPG” na stronie 1410

Składnia

(HCONN, HSUB, ACTION, SUBROPT, CMPCOD, REASON) MQSUBRQ

Użycie notatek

Poniższe uwagi dotyczące użycia mają zastosowanie do SRAPUB:

1. Jeśli wykonanie tej komendy zakończy się pomyślnie, zachowane publikacje zgodne z określoną subskrypcją zostaną wysłane do subskrypcji i mogą zostać odebrane za pomocą komendy MQGET lub MQCB przy użyciu komendy HOBJ zwróconej w oryginalnej komendzie MQSUB, która utworzyła subskrypcję.
2. Jeśli temat zasubskrybowany przez pierwotną komendę MQSUB, która utworzyła subskrypcję, zawiera znak wieloznaczny, może zostać wysłana więcej niż jedna zachowana publikacja. Liczba publikacji wysłanych w wyniku tego wywołania jest rejestrowana w polu *SRNMP* w strukturze SBROPT.
3. Jeśli komenda zostanie zakończona z kodem przyczyny RC2437 , oznacza to, że nie zachowano żadnych publikacji dla podanego tematu.
4. Jeśli komenda zostanie zakończona z kodem przyczyny RC2525 lub RC2526 , oznacza to, że istnieją obecnie zachowane publikacje dla określonego tematu, ale wystąpił błąd, który oznacza, że nie można ich dostarczyć.
5. Aplikacja musi mieć bieżącą subskrypcję tematu, zanim będzie mogła wykonać to wywołanie. Jeśli subskrypcja została utworzona w poprzedniej instancji aplikacji i nie jest dostępny poprawny uchwyt subskrypcji, aplikacja musi najpierw wywołać MQSUB z opcją SORES, aby uzyskać uchwyt do użycia w tym wywołaniu.
6. Publikacje są wysyłane do miejsca docelowego, które jest zarejestrowane do użytku z bieżącą subskrypcją tej aplikacji. Jeśli publikacje powinny zostać wysłane w innym miejscu, należy najpierw zmienić subskrypcję za pomocą wywołania MQSUB z opcją SOALT.

Parametry

Wywołanie MQSUBRQ ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *HCONN* została zwrócona przez poprzednie wywołanie MQCONN lub MQCONNX.

W przypadku aplikacji z/OS for CICS można pominąć wywołanie MQCONN i podać następującą wartość parametru *HCONN*:

HHCDEFH
Domyślny uchwyt połączenia.

HSUB (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten uchwyt reprezentuje subskrypcję, dla której ma zostać zażądana aktualizacja. Wartość *HSUB* została zwrócona z poprzedniego wywołania MQSUB.

ACTION (10-cyfrowa liczba całkowita ze znakiem)-wejście

Ten parametr steruje konkretnym działaniem, które jest żądane dla subskrypcji. Należy określić jeden (i tylko jeden) z następujących elementów:

SRAPUB

To działanie żąda wysłania publikacji aktualizacji dla określonego tematu. Ta opcja jest zwykle używana, jeśli subskrybent określił opcję SOPUBR w wywołaniu MQSUB podczas tworzenia subskrypcji. Jeśli menedżer kolejek ma zachowaną publikację dla tematu, jest ona wysyłana do subskrybenta. Jeśli nie, wywołanie nie powiedzie się. Jeśli do aplikacji zostanie wysłana zachowana publikacja, jest to wskazywane przez właściwość komunikatu MQIsRetained tej publikacji.

Ponieważ temat w istniejącej subskrypcji reprezentowany przez parametr **HSUB** może zawierać znaki wieloznaczne, subskrybent może otrzymać wiele zachowanych publikacji.

SBROPT (MQSRO)-wejście/wyjście

Te opcje sterują działaniem programu MQSUBRQ. Szczegółowe informacje na ten temat zawiera sekcja “MQSRO-opcje żądania subskrypcji” na stronie 607 .

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia; jest to jeden z następujących kodów:

CKOK
Pomyślne zakończenie

CWARN
Ostrzeżenie (częściowe zakończenie)

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się

Przyczyna (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny, który kwalifikuje się jako *CMPCOD*.

Jeśli *CPMCOD* to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CPMCOD* to CCFAIL:

RC2298
2298 (X'08FA') Żądana funkcja nie jest dostępna w bieżącym środowisku.

RC2437
2437 (X'0985 ') Dla tego tematu nie są obecnie przechowywane żadne zachowane publikacje.

RC2046
Parametr lub pole opcji 2046 (X'07FE') zawiera niepoprawne opcje lub kombinację niepoprawnych opcji.

RC2161
2161 (X'0871 ') wygaszanie menedżera kolejek

RC2438
2438 (X'0986 ') W wywołaniu MQSUBRQ opcje żądania subskrypcji MQSRO są niepoprawne.

Deklaracja RPG

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQSUBRQ(HCONN : HSUB : ACTION :
C                                       SBROPT : CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQSUBRQ          PR                  EXTPROC('MQSUBRQ')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Subscription handle
D HSUB                          10I 0 VALUE
D* Action requested on the subscription
D ACTION                        10I 0 VALUE
D* Subscription Request Options
D SBROPT                        16A
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CompCode
D REASON                        10I 0
```

IBM i Atrybuty obiektów w systemie IBM i

W tej kolekcji tematów wymieniono tylko te obiekty IBM MQ , które mogą być przedmiotem wywołania funkcji MQINQ, oraz podano szczegółowe informacje o atrybutach, do których można tworzyć zapytania, i selektorach, które mają być używane.

Atrybuty kolejek

Ten temat zawiera informacje o różnych typach definicji kolejek i atrybutach obsługiwanych przez każdą z nich.

Typy kolejek: Menedżer kolejek obsługuje następujące typy definicji kolejek:

Kolejka lokalna
Jest to kolejka fizyczna, w której przechowywane są komunikaty. Kolejka istnieje w lokalnym menedżerze kolejek.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu i usuwać je z nich. Wartością atrybutu kolejki **QType** jest QTLOC.

Kolejka współużytkowana
Jest to kolejka fizyczna, w której przechowywane są komunikaty. Kolejka istnieje we współużytkowanym repozytorium dostępnym dla wszystkich menedżerów kolejek należących do grupy współużytkowania kolejek, do której należy repozytorium współużytkowane.

Aplikacje połączone z dowolnym menedżerem kolejek w grupie współużytkowania kolejek mogą umieszczać komunikaty w kolejkach tego typu i usuwać je z nich. Takie kolejki są w rzeczywistości takie same, jak kolejki lokalne. Wartością atrybutu kolejki **QType** jest QTLOC.

- Kolejki współużytkowane są obsługiwane tylko w systemie z/OS.

Kolejka klastra
Jest to kolejka fizyczna, w której przechowywane są komunikaty. Kolejka istnieje w menedżerze kolejek lokalnych lub w co najmniej jednym z menedżerów kolejek należących do tego samego klastra co menedżer kolejek lokalnych.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach tego typu, niezależnie od położenia kolejki. Jeśli instancja kolejki istnieje w menedżerze kolejek lokalnych, kolejka zachowuje się tak samo, jak kolejka lokalna, a aplikacje połączone z menedżerem kolejek lokalnych mogą usuwać komunikaty z kolejki. Wartością atrybutu kolejki **QType** jest QTCLUS.

Kolejka aliasowa
Nie jest to kolejka fizyczna-jest to nazwa alternatywna dla kolejki lokalnej. Nazwa kolejki lokalnej, na którą alias jest tłumaczony, jest częścią definicji kolejki aliasowej.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w kolejkach aliasowych i usuwać je z nich-komunikaty są umieszczane w kolejce lokalnej, na którą alias jest tłumaczony, oraz usuwane z tej kolejki. Wartością atrybutu kolejki **QType** jest QTALS.

Kolejka zdalna
Nie jest to kolejka fizyczna-jest to lokalna definicja kolejki, która istnieje w zdalnym menedżerze kolejek. Definicja lokalna kolejki zdalnej zawiera informacje, które informują menedżera kolejek lokalnych o sposobie kierowania komunikatów do menedżera kolejek zdalnych.

Aplikacje połączone z lokalnym menedżerem kolejek mogą umieszczać komunikaty w zdalnych kolejkach-komunikaty są umieszczane w lokalnej kolejce transmisji używanej do kierowania komunikatów do zdalnego menedżera kolejek. Aplikacje nie mogą usuwać komunikatów z kolejek zdalnych. Wartością atrybutu kolejki **QType** jest QTREM.

Definicja kolejki zdalnej może być również używana do:

- Używanie aliasów w kolejkach odpowiedzi

  W tym przypadku nazwa definicji jest nazwą kolejki odpowiedzi. Więcej informacji na ten temat zawiera sekcja Definicje aliasów kolejek odpowiedzi.
- Używanie aliasów menedżera kolejek

  W tym przypadku nazwa definicji jest aliasem menedżera kolejek, a nie nazwą kolejki. Więcej informacji na ten temat zawiera sekcja Definicje aliasów menedżera kolejek.

Kolejka modelowa
Nie jest to kolejka fizyczna-jest to zestaw atrybutów kolejki, na podstawie których można utworzyć kolejkę lokalną.

Komunikaty nie mogą być przechowywane w kolejkach tego typu.

Niektóre atrybuty kolejki mają zastosowanie do wszystkich typów kolejek; inne atrybuty kolejki mają zastosowanie tylko do niektórych typów kolejek. Typy kolejek, do których atrybut ma zastosowanie, są oznaczone znakiem "X" w tabeli Tabela 754 na stronie 1412 i kolejnych tabelach.

Tabela 754 na stronie 1412 zawiera podsumowanie atrybutów specyficznych dla kolejek. Atrybuty są opisane w porządku alfabetycznym.

Nazwy atrybutów wyświetlane w tabeli są nazwami używanymi w wywołaniach MQINQ i MQSET. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Szczegółowe informacje na ten temat zawiera sekcja Komendy MQSC .

W poniższej tabeli kolumny mają zastosowanie w następujący sposób:

- Kolumna dla kolejek lokalnych dotyczy również kolejek współużytkowanych.
- Kolumna dla kolejek modelowych wskazuje, które atrybuty są dziedziczone przez kolejkę lokalną utworzoną z kolejki modelowej.
- Kolumna dla kolejek klastra wskazuje atrybuty, które można uzyskać, gdy kolejka klastra jest otwierana tylko dla zapytań lub dla zapytań i danych wyjściowych. Jeśli kolejka klastra jest otwarta w celu wykonania zapytania plus co najmniej jedna z wartości wejściowych, przeglądania lub ustawiania, zamiast tego stosowana jest kolumna dla kolejek lokalnych.

Tabela 754. Atrybuty kolejek						
Atrybut	**Opis**	**Lokalna**	**Model**	**Alias**	**Zdalna**	**Klaster**
AlterationDate	Data ostatniej zmiany definicji	X		X	X	
AlterationTime	Czas ostatniej zmiany definicji	X		X	X	
BackoutRequeueQName	Nadmierna ilość kolejki wycofanych komunikatów	X	X			
BackoutThreshold	Próg wycofania	X	X			
BaseQName	Nazwa kolejki, na którą alias jest tłumaczony			X		
ClusterChannelNazwa	Nazwa kanału nadawczego klastra	✓	✓			
ClusterName	Nazwa klastra, do którego należy kolejka	X		X	X	
ClusterNamelist	Nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy kolejka	X		X	X	
CreationDate	Data utworzenia kolejki	X				
CreationTime	Czas utworzenia kolejki	X				
CurrentQDepth	Bieżąca głębokość kolejki	X				
DefBind	Domyślne łączenie	X		X	X	X
DefinitionType	Typ definicji kolejki.	X	X			
DefInputOpenOption	Domyślna opcja otwarcia wejścia	X	X			
DefPersistence	Domyślna trwałość komunikatu	X	X	X	X	X
DefPriority	Domyślny priorytet komunikatu	X	X	X	X	X
DistLists	Obsługa listy dystrybucyjnej	X	X			
HardenGetHardenGet	Czy zachować dokładną liczbę wycofań	X	X			
InhibitGet	Określa, czy operacje pobierania dla kolejki są dozwolone	X	X	X		

Tabela 754. Atrybuty kolejek (kontynuacja)

Atrybut	Opis	Lokalna	Model	Alias	Zdalna	Klaster
InhibitPut	Określa, czy operacje umieszczania dla kolejki są dozwolone	X	X	X	X	X
InitiationQName	Nazwa kolejki inicjującej	X	X			
MaxMsgdługość komunikatu	Maksymalna długość komunikatu w bajtach	X	X			
MaxQDepth	Maksymalna głębokość kolejki	X	X			
MediaLog	Tożsamość najstarszego przydziału dziennika (lub najstarszego dziennika w systemie IBM i) potrzebne do odtwarzania nośników określonej kolejki	✓	✓			
MsgDeliverySekwencja	Kolejność dostarczania komunikatów	X	X			
LiczbaOpenInput	Liczba operacji otwierania do wprowadzania	X				
Liczba:OpenOutput	Liczba operacji otwarcia dla danych wyjściowych	X				
ProcessName	Nazwa procesu	X	X			
ZdarzenieQDepthHigh	Określa, czy generowane są zdarzenia nadmiaru kolejki	X	X			
LimitQDepthHigh	Górny limit głębokości kolejki	X	X			
ZdarzenieQDepthLow	Określa, czy generowane są zdarzenia niedoboru kolejki	X	X			
QDepthLowLimit	Dolny limit głębokości kolejki	X	X			
ZdarzenieQDepthMax	Określa, czy generowane są zdarzenia zapełnienia kolejki	X	X			
QDesc (Opis kolejek)	Opis kolejki	X	X	X	X	X

Tabela 754. Atrybuty kolejek (kontynuacja)						
Atrybut	**Opis**	**Lokalna**	**Model**	**Alias**	**Zdalna**	**Klaster**
Nazwa QName	Nazwa kolejki	X		X	X	X
QServiceInterval	Miejsce docelowe dla odstępu czasu usługi kolejki	X	X			
ZdarzenieQServiceInterval	Określa, czy generowane są zdarzenia wysokiego, czy też prawidłowego odstępu czasu usługi.	X	X			
QTYPE	Typ kolejki	X		X	X	X
RemoteQMgrNazwa	Nazwa zdalnego menedżera kolejek				X	
RemoteQName	Nazwa kolejki zdalnej				X	
RetentionInterval	Interwał przechowywania	X	X			
SCOPE	Określa, czy pozycja dla kolejki istnieje również w katalogu komórki	X		X	X	
Możliwość współużytkowania	Możliwość współużytkowania kolejki	X	X			
TriggerControl	Kontrola wyzwalacza	X	X			
TriggerData	Dane wyzwalacza	X	X			
TriggerDepth	Wyzwalacz uruchamiany zapełnieniem	X	X			
TriggerMsgPriorytet	Próg priorytetu komunikatu dla wyzwalacza.	X	X			
TriggerType	Typ wyzwalacza	X	X			
Użycie	Wykorzystanie kolejki	X	X			
XmitQName	Nazwa kolejki transmisji				X	

IBM i ***AlterationDate (12-bajtowy łańcuch znaków) w systemie IBM i***

Data ostatniej zmiany definicji.

Tabela 755. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów (na przykład 1992-09-23¬¬, gdzie ¬¬ reprezentuje dwa znaki odstępu).

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) zmieniają się w miarę działania menedżera kolejek. Zmiany tych atrybutów nie mają wpływu na *AlterationDate*.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNDATE.

IBM i *AlterationTime (8-bajtowy łańcuch znaków) w systemie IBM i*

Czas ostatniej zmiany definicji.

Tabela 756. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS z użyciem zegara 24-godzinnego, z zerem wiodącym, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20). Jest to czas miejscowy.

Wartości niektórych atrybutów (na przykład *CurrentQDepth*) zmieniają się w miarę działania menedżera kolejek. Zmiany tych atrybutów nie mają wpływu na *AlterationTime*.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNTIME.

IBM i *BackoutRequeueQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nadmierna liczba wycofanych komunikatów w kolejce.

Tabela 757. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Aplikacje działające w systemie WebSphere Application Server oraz aplikacje korzystające z narzędzi IBM MQ Application Server Facilities używają tego atrybutu do określenia miejsca, w którym powinny być wyświetlane wycofane komunikaty. W przypadku wszystkich innych aplikacji, oprócz zezwolenia na odpytywanie o jej wartość, menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora CABRQN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *BackoutThreshold (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Próg wycofania.

Tabela 758. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Aplikacje działające w systemie WebSphere Application Server oraz aplikacje korzystające z narzędzi IBM MQ Application Server Facilities używają tego atrybutu do określenia, czy komunikat powinien zostać wycofany. W przypadku wszystkich innych aplikacji, oprócz zezwolenia na odpytywanie o jej wartość, menedżer kolejek nie podejmuje żadnych działań w oparciu o wartość atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora IABTHR z wywołaniem MQINQ.

IBM i *BaseQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki, na którą alias jest tłumaczony.

Tabela 759. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
		X		

Jest to nazwa kolejki, która jest zdefiniowana dla lokalnego menedżera kolejek. Więcej informacji na temat nazw kolejek zawiera opis pola *ODON* w dokumencie MQOD. Kolejka może być jednego z następujących typów:

QTLOC,
Kolejka lokalna.

QTREM
Lokalna definicja kolejki zdalnej.

QTCLUS
Kolejka klastra.

Aby określić wartość tego atrybutu, należy użyć selektora CABASQ z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *BaseType (struktura parametru liczby całkowitej) w systemie IBM i*

Typ obiektu, na który alias jest tłumaczony.

Tabela 760. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
		X		

Ten atrybut może mieć jedną z następujących wartości:

OTQ
Podstawowym typem obiektu jest kolejka

OTTOP
Podstawowy typ obiektu jest tematem

IBM i *CFStrucName (12-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa struktury narzędzia CF.

Tabela 761. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to nazwa struktury narzędzia CF, w której przechowywane są komunikaty w kolejce. Pierwszy znak nazwy należy do zakresu od A do Z, a pozostałe znaki należą do zakresu od A do Z, od 0 do 9 lub są puste.

Pełną nazwę struktury w narzędziu CF uzyskuje się przez dodanie do wartości atrybutu menedżera kolejek **QSGName** wartości atrybutu kolejki **CFStrucName** .

Ten atrybut ma zastosowanie tylko do kolejek współużytkowanych; jest ignorowany, jeśli parametr *QSGDisp* nie ma wartości QSGDSH.

Aby określić wartość tego atrybutu, należy użyć selektora CACFSN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNCFSN.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

ClusterChannelNazwa (20-bajtowy łańcuch znaków)

ClusterChannelNazwa to nazwa ogólna kanałów nadawczych klastra, które używają tej kolejki jako kolejki transmisji. Atrybut określa, które kanały nadawcze klastra wysyłają komunikaty do kanału odbiorczego klastra z tej kolejki transmisji klastra.

Tabela 762. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Domyślna konfiguracja menedżera kolejek dotyczy wszystkich kanałów nadawczych klastra mających wysyłać komunikaty z pojedynczej kolejki transmisji SYSTEM.CLUSTER.TRANSMIT.QUEUE. Konfigurację domyślną można zmienić, modyfikując, zmieniając atrybut menedżera kolejek **DefClusterXmitQueueType**. Wartością domyślną tego atrybutu jest SCTQ. Wartość tę można zmienić na CHANNEL. Jeśli atrybut **DefClusterXmitQueueType** zostanie ustawiony na wartość CHANNEL, dla każdego kanału nadawczego klastra domyślnie zostanie użyta konkretna kolejka transmisji klastra SYSTEM.CLUSTER.TRANSMIT.*ChannelName*.

Kanał nadawczy klastra dla atrybutu ClusterChannelName kolejki transmisji można również ustawić ręcznie. Komunikaty przeznaczone dla menedżera kolejek połączonego kanałem nadawczym klastra są przechowywane w kolejce transmisji identyfikującej kanał nadawczy klastra. Nie są one przechowywane w domyślnej kolejce transmisji klastra. Jeśli dla atrybutu ClusterChannelName zostaną ustawione wartości puste, po zrestartowaniu kanału zostanie on przełączony na domyślną kolejkę transmisji klastra. Kolejka domyślna to SYSTEM.CLUSTER.TRANSMIT.*ChannelName* lub SYSTEM.CLUSTER.TRANSMIT.QUEUE, w zależności od wartości atrybutu DefClusterXmitQueueType menedżera kolejek.

Określając gwiazdki ("*") w programie **ClusterChannelName**, można powiązać kolejkę transmisji z zestawem kanałów nadawczych klastra. Gwiazdki mogą znajdować się na początku, na końcu lub na dowolnej liczbie miejsc w środku łańcucha nazwy kanału. **ClusterChannelName** o długości ograniczonej do 20 znaków: MQ_CHANNEL_NAME_LENGTH.

IBM i *ClusterName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa klastra, do którego należy kolejka.

Tabela 763. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	

Jest to nazwa klastra, do którego należy kolejka. Jeśli kolejka należy do więcej niż jednego klastra, parametr *ClusterNamelist* określa nazwę obiektu listy nazw, który identyfikuje klastry, a parametr *ClusterName* jest pusty. Co najmniej jedna z wartości *ClusterName* i *ClusterNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CACLN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNCLUN.

IBM i *ClusterNamelist (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy kolejka.

Tabela 764. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	

Jest to nazwa obiektu listy nazw zawierającego nazwy klastrów, do których należy ta kolejka. Jeśli kolejka należy tylko do jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć parametru *ClusterName* , aby określić nazwę klastra, w którym to przypadku wartość *ClusterNamelist* jest pusta. Co najmniej jedna z wartości *ClusterName* i *ClusterNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CACLNL z wywołaniem MQINQ. Długość tego atrybutu jest określona przez numer LNNLN.

IBM i *CreationDate (12-bajtowy łańcuch znaków) w systemie IBM i*

Data utworzenia kolejki.

Tabela 765. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X				

Data utworzenia kolejki. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów (na przykład 1992-09-23¬¬, gdzie ¬¬ reprezentuje dwa znaki odstępu).

- W systemie IBM idata utworzenia kolejki może różnić się od daty utworzenia bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora CACRTD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNCRTD.

IBM i *CreationTime (8-bajtowy łańcuch znaków) w systemie IBM i*

Czas utworzenia kolejki.

Tabela 766. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X				

Jest to czas utworzenia kolejki. Format godziny to HH.MM.SS z użyciem zegara 24-godzinnego, z zerem wiodącym, jeśli godzina jest mniejsza niż 10 (na przykład 09.10.20). Jest to czas miejscowy.

- W systemie IBM iczas utworzenia kolejki może różnić się od czasu utworzenia bazowej jednostki systemu operacyjnego (pliku lub przestrzeni użytkownika), która reprezentuje kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora CACRTT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNCRTT.

IBM i *CurrentQDepth (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Bieżące zapełnienie kolejki.

Tabela 767. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

To jest liczba komunikatów znajdujących się aktualnie w kolejce. Wartość ta jest zwiększana podczas wywołania MQPUT i wycofywania wywołania MQGET. Jest on zmniejszany podczas wykonywania wywołania MQGET bez przeglądania i podczas wycofywania wywołania MQPUT. W rezultacie licznik uwzględnia komunikaty, które zostały umieszczone w kolejce w ramach jednostki pracy, ale które nie zostały jeszcze zatwierdzone, nawet jeśli nie zostały zakwalifikowane do pobrania przez wywołanie MQGET. Podobnie wyklucza komunikaty, które zostały pobrane w ramach jednostki pracy za pomocą wywołania MQGET, ale które nie zostały jeszcze zatwierdzone.

Liczba ta obejmuje również komunikaty, które przekroczyli czas utraty ważności, ale nie zostały jeszcze odrzucone, chociaż nie kwalifikują się do pobrania. Patrz opis pola *MDEXP* w sekcji "MQMD (deskryptor komunikatu) w systemie IBM i" na stronie 1141.

Zarówno przetwarzanie jednostek pracy, jak i segmentacja komunikatów mogą spowodować przekroczenie przez `CurrentQDepth` wartości `MaxQDepth`. Nie ma to jednak wpływu na możliwość pobierania komunikatów- *wszystkie* komunikaty w kolejce mogą być pobierane za pomocą wywołania MQGET w normalny sposób.

Wartość tego atrybutu zmienia się w miarę działania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IACDEP z wywołaniem MQINQ.

IBM i *DefBind (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Powiązanie domyślne.

Tabela 768. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Ten atrybut jest powiązaniem domyślnym używanym w przypadku, gdy w wywołaniu MQOPEN określono parametr OOBNDQ, a kolejka jest kolejką klastra. DefBind może mieć jedną z następujących wartości:

BNDOPN
Powiązanie naprawione przez wywołanie MQOPEN.

NIE BNDNOT
Powiązanie nie jest stałe.

BNDGRP
Powiązanie nie jest naprawione przez wywołanie MQOPEN, ale jest naprawione w MQPUT dla wszystkich komunikatów w grupie logicznej.

Aby określić wartość tego atrybutu, należy użyć selektora IADBND z wywołaniem MQINQ.

IBM i *DefinitionType (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Typ definicji kolejki.

Tabela 769. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Wskazuje sposób zdefiniowania kolejki. Jest to jedna z następujących wartości:

QDPRE,
Predefiniowana kolejka trwała.

Kolejka jest kolejką trwałą utworzoną przez administratora systemu; tylko administrator systemu może ją usunąć.

Predefiniowane kolejki są tworzone za pomocą komendy DEFINE MQSC i można je usunąć tylko za pomocą komendy DELETE MQSC. Nie można tworzyć predefiniowanych kolejek na podstawie kolejek modelowych.

Komendy mogą być wydawane przez operatora lub przez autoryzowanego użytkownika wysyłającego komunikat komendy do kolejki wejściowej komend (patrz atrybut **CommandInputQName** opisany w sekcji "Atrybuty menedżera kolejek w systemie IBM i" na stronie 1443).

KDPERM
Dynamicznie definiowana kolejka trwała.

Kolejka jest kolejką trwałą, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określonej w deskryptorze obiektu MQOD. Definicja kolejki modelowej miała wartość QDPERM dla atrybutu **DefinitionType** .

Ten typ kolejki można usunąć za pomocą wywołania MQCLOSE. Więcej informacji na temat zawiera sekcja "MQCLOSE (Close object-Zamknij obiekt) w systemie IBM i" na stronie 1307.

Wartością atrybutu **QSGDisp** dla trwałej kolejki dynamicznej jest QSGDQM.

QDTEMP,
Dynamicznie definiowana kolejka tymczasowa.

Kolejka jest kolejką tymczasową, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskryptorze obiektu MQOD. Definicja kolejki modelowej miała wartość QDTEMP dla atrybutu **DefinitionType** .

Ten typ kolejki jest automatycznie usuwany przez wywołanie MQCLOSE po jego zamknięciu przez aplikację, która go utworzyła.

Wartością atrybutu **QSGDisp** dla tymczasowej kolejki dynamicznej jest QSGDQM.

QDSHAR,
Dynamicznie definiowana kolejka współużytkowana.

Kolejka jest współużytkowaną kolejką trwałą, która została utworzona przez aplikację wywołującą wywołanie MQOPEN z nazwą kolejki modelowej określoną w deskryptorze obiektu MQOD. Definicja kolejki modelowej miała wartość QDSHAR dla atrybutu **DefinitionType** .

Ten typ kolejki można usunąć za pomocą wywołania MQCLOSE. Więcej informacji na temat zawiera sekcja "MQCLOSE (Close object-Zamknij obiekt) w systemie IBM i" na stronie 1307.

Wartością atrybutu **QSGDisp** dla współużytkowanej kolejki dynamicznej jest QSGDSH.

Ten atrybut w definicji kolejki modelowej nie wskazuje sposobu zdefiniowania kolejki modelowej, ponieważ kolejki modelowe są zawsze predefiniowane. Zamiast tego wartość tego atrybutu w kolejce modelowej jest używana do określenia wartości *DefinitionType* dla każdej kolejki dynamicznej utworzonej na podstawie definicji kolejki modelowej przy użyciu wywołania MQOPEN.

Aby określić wartość tego atrybutu, należy użyć selektora IADEFT z wywołaniem MQINQ.

IBM i *DefInputOpenOption (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Domyślna opcja otwarcia wejścia.

Tabela 770. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to domyślny sposób otwierania kolejki dla danych wejściowych. Ma ona zastosowanie, jeśli podczas otwierania kolejki w wywołaniu MQOPEN podano opcję OOINPQ. Może mieć jedną z następujących wartości:

OOOINPX
Otwórz kolejkę, aby pobrać komunikaty z dostępem na wyłączność.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie kończy się niepowodzeniem z kodem przyczyny RC2042 , jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację do wprowadzania danych dowolnego typu (OOINPS lub OOINPX).

OOOINPS
Otwórz kolejkę, aby pobrać komunikaty z dostępem współużytkowanym.

Kolejka jest otwierana do użytku z kolejnymi wywołaniami MQGET. Wywołanie może zakończyć się pomyślnie, jeśli kolejka jest obecnie otwarta przez tę lub inną aplikację z wartością OOINPS, ale zakończy się niepowodzeniem z kodem przyczyny RC2042 , jeśli kolejka jest obecnie otwarta z wartością OOINPX.

Aby określić wartość tego atrybutu, należy użyć selektora IADINP z wywołaniem MQINQ.

IBM i *DefPersistence (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Domyślna trwałość komunikatu.

Tabela 771. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X	X	X	X

Jest to domyślna trwałość komunikatów w kolejce. Ma ona zastosowanie, jeśli podczas umieszczania komunikatu w deskryptorze komunikatu określono PEQDEF.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślna trwałość jest pobierana z wartości tego atrybutu w *pierwszej* definicji w ścieżce w czasie wywołania MQPUT lub MQPUT1 . Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład *DefXmitQName*)

Może mieć jedną z następujących wartości:

PEPER,
Komunikat jest trwały.

Oznacza to, że komunikat przetrwa awarie systemu i restarty menedżera kolejek. Nie można umieścić trwałych komunikatów w:

- Tymczasowe kolejki dynamiczne
- Kolejki współużytkowane

Komunikaty trwałe mogą być umieszczane w trwałych kolejkach dynamicznych oraz w predefiniowanych kolejkach.

PENPER
Komunikat nie jest trwały.

Oznacza to, że komunikat zwykle nie przetrwa awarii systemu ani restartów menedżera kolejek. Ma to zastosowanie nawet wtedy, gdy podczas restartowania menedżera kolejek w pamięci dyskowej zostanie znaleziona nienaruszona kopia komunikatu.

W przypadku kolejek współużytkowanych komunikaty nietrwałe *pozostają* po restarcie menedżerów kolejek w grupie współużytkowania kolejek, ale nie są zachowywane awarie narzędzia CF używanego do przechowywania komunikatów we współużytkowanych kolejkach.

W tej samej kolejce mogą istnieć zarówno komunikaty trwałe, jak i nietrwałe.

Aby określić wartość tego atrybutu, należy użyć selektora IADPER z wywołaniem MQINQ.

IBM i *DefPriority (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Domyślny priorytet komunikatu.

Tabela 772. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jest to domyślny priorytet komunikatów w kolejce. Ma to zastosowanie, jeśli parametr PRQDEF jest określony w deskryptorze komunikatu, gdy komunikat jest umieszczany w kolejce.

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, domyślny priorytet dla komunikatu jest pobierany z wartości tego atrybutu w *pierwszej* definicji w ścieżce w czasie operacji umieszczania (put). Może to być:

- Kolejka aliasowa
- Kolejka lokalna
- Lokalna definicja kolejki zdalnej
- Alias menedżera kolejek
- Kolejka transmisji (na przykład *DefXmitQName*)

Sposób umieszczania komunikatu w kolejce zależy od wartości atrybutu **MsgDeliverySequence** kolejki:

- Jeśli atrybut **MsgDeliverySequence** ma wartość MSPRIO, pozycja logiczna, w której komunikat jest umieszczany w kolejce, zależy od wartości pola *MDPRI* w deskryptorze komunikatu.
- Jeśli atrybut **MsgDeliverySequence** ma wartość MSFIFO, komunikaty są umieszczane w kolejce tak, jakby miały priorytet równy *DefPriority* rozstrzygniętej kolejki, niezależnie od wartości pola *MDPRI* w deskryptorze komunikatu. Jednak pole *MDPRI* zachowuje wartość określoną przez aplikację, która umieściła komunikat. Więcej informacji na ten temat zawiera opis atrybutu **MsgDeliverySequence** w sekcji “Atrybuty kolejek” na stronie 1410 .

Priorytety należą do zakresu od 0 (najniższy) do *MaxPriority* (najwyższy); patrz atrybut **MaxPriority** opisany w sekcji “Atrybuty menedżera kolejek w systemie IBM i” na stronie 1443.

Aby określić wartość tego atrybutu, należy użyć selektora IADPRI z wywołaniem MQINQ.

IBM i *DefReadz wyprzedzeniem (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa domyślne zachowanie odczytu z wyprzedzeniem dla nietrwałych komunikatów dostarczanych do klienta.

Tabela 773. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X	X		

DefReadMożna ustawić jedną z następujących wartości:

RAHNO
Komunikaty nietrwałe nie są wysyłane z wyprzedzeniem do klienta, zanim aplikacja ich zażąda. Jeśli działanie klienta zostanie zakończone nieprawidłowo, może zostać utracony maksymalnie jeden komunikat nietrwały.

RAHYES
Nietrwałe komunikaty są wysyłane do klienta przed zażądaniem ich przez aplikację. Nietrwałe komunikaty mogą zostać utracone, jeśli klient zostanie zakończony nieprawidłowo lub jeśli nie odbierze wszystkich wysłanych komunikatów.

RAD
Odczyt z wyprzedzeniem nietrwałych komunikatów nie jest włączony dla tej kolejki. Komunikaty nie są wysyłane do klienta z wyprzedzeniem, niezależnie od tego, czy aplikacja kliencka żąda odczytu z wyprzedzeniem.

Aby określić wartość tego atrybutu, należy użyć selektora IADRAH z wywołaniem MQINQ.

IBM i *DefPResp (10-cyfrowa liczba całkowita ze znakiem) on IBM i*

Atrybut domyślnego typu odpowiedzi umieszczania (DEFPRESP) definiuje wartość używaną przez aplikacje, gdy dla typu PutResponsew ramach MQPMO ustawiono wartość PMRASQ. Ten atrybut jest poprawny dla wszystkich typów kolejek.

Tabela 774. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X	X	X	X

Może mieć jedną z następujących wartości:

SYNCHRONICZNY
Operacja put jest wykonywana synchronicznie i zwraca odpowiedź.

ASYNCHRONICZNY
Operacja put jest wykonywana asynchronicznie, zwracając podzbiór pól MQMD.

Aby określić wartość tego atrybutu, należy użyć selektora IADPRT z wywołaniem MQINQ.

IBM i *DistLists (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Obsługa listy dystrybucyjnej.

Tabela 775. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Wskazuje, czy komunikaty listy dystrybucyjnej mogą być umieszczane w kolejce. Atrybut jest ustawiany przez agent kanału komunikatów (MCA) w celu poinformowania lokalnego menedżera kolejek, czy menedżer kolejek na drugim końcu kanału obsługuje listy dystrybucyjne. Ten drugi menedżer kolejek (nazywany partnerskim menedżerem kolejek) jest kolejnym, który odbiera komunikat po usunięciu go z lokalnej kolejki transmisji przez wysyłający agent MCA.

Atrybut jest ustawiany przez wysyłający agent MCA za każdym razem, gdy nawiązuje on połączenie z odbierającym agentem MCA w partnerskim menedżerze kolejek. W ten sposób wysyłający agent MCA może spowodować, że lokalny menedżer kolejek umieści w kolejce transmisji tylko te komunikaty, które partnerujący menedżer kolejek może przetworzyć poprawnie.

Ten atrybut jest przeznaczony przede wszystkim do użytku z kolejkami transmisji, ale opisane przetwarzanie jest wykonywane niezależnie od użycia zdefiniowanego dla kolejki (patrz atrybut **Usage**).

Może mieć jedną z następujących wartości:

Obsługa DLSUPP
Obsługiwane listy dystrybucyjne.

Oznacza to, że komunikaty z listy dystrybucyjnej mogą być przechowywane w kolejce i przesyłane do partnerującego menedżera kolejek w tej postaci. Zmniejsza to ilość przetwarzania wymaganego do wysłania komunikatu do wielu miejsc docelowych.

DLNSUP
Listy dystrybucyjne nie są obsługiwane.

Oznacza to, że komunikaty listy dystrybucyjnej nie mogą być składowane w kolejce, ponieważ partnerski menedżer kolejek nie obsługuje list dystrybucyjnych. Jeśli aplikacja umieszcza komunikat listy dystrybucyjnej i komunikat ten ma zostać umieszczony w tej kolejce, menedżer kolejek dzieli komunikat listy dystrybucyjnej i umieszcza zamiast tego poszczególne komunikaty w kolejce. Zwiększa to ilość przetwarzania wymaganego do wysłania komunikatu do wielu miejsc docelowych, ale zapewnia, że komunikaty będą poprawnie przetwarzane przez współpracujący menedżer kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IADIST z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *HardenGetBackout (10-cyfrowa liczba całkowita ze znakiem) on IBM i*

Określa, czy zachować dokładną liczbę wycofań.

Tabela 776. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Dla każdego komunikatu jest zachowywana liczba pobrań komunikatu przez wywołanie MQGET w obrębie jednostki pracy, a następnie ta jednostka pracy została wycofana. Ten licznik jest dostępny w polu *MDBOC* w deskryptorze komunikatu po zakończeniu wywołania MQGET.

Liczba wycofanych komunikatów przetrwa, gdy menedżer kolejek zostanie zrestartowany. Jednak aby zapewnić dokładność licznika, informacje muszą być "wzmocnione" (zapisane na dysku lub innym trwałym urządzeniu pamięci masowej) za każdym razem, gdy komunikat jest pobierany przez wywołanie MQGET w ramach jednostki pracy dla tej kolejki. Jeśli ta czynność nie zostanie wykonana, a wystąpi niepowodzenie menedżera kolejek wraz z wycofaniem wywołania MQGET, licznik może nie zostać zwiększony.

Zwiększenie ilości informacji dla każdego wywołania MQGET w jednostce pracy powoduje jednak obniżenie wydajności, a atrybut **HardenGetBackout** powinien być ustawiony na wartość QABH tylko wtedy, gdy liczba ta musi być dokładna.

- W systemie IBM iliczba wycofanych komunikatów jest zawsze zachowana, niezależnie od ustawienia tego atrybutu.

Dozwolone są następujące wartości:

QABH
Liczba wycofań zapamiętanych.

Wzmacnianie jest używane w celu zapewnienia, że liczba wycofanych komunikatów w tej kolejce jest dokładna.

QABNH
Liczba wycofań może nie zostać zapamiętana.

Wzmacnianie nie jest używane w celu upewnienia się, że liczba wycofanych komunikatów w tej kolejce jest dokładna. W związku z tym liczba ta może być niższa niż powinna.

Aby określić wartość tego atrybutu, należy użyć selektora IAHGB z wywołaniem MQINQ.

IBM i *InhibitGet (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy operacje pobierania dla tej kolejki są dozwolone.

Tabela 777. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X		

Jeśli kolejka jest kolejką aliasową, operacje pobierania muszą być dozwolone zarówno dla kolejki aliasowej, jak i dla kolejki podstawowej w czasie wykonywania operacji pobierania, aby wywołanie MQGET powiodło się. Jest to jedna z następujących wartości:

QAGETI
Operacje pobierania są zablokowane.

Wywołania MQGET nie powiodły się z kodem przyczyny RC2016. Obejmuje to wywołania MQGET, które określają GMBRWF lub GMBRWN.

Uwaga: Jeśli wywołanie MQGET działające w obrębie jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu **InhibitGet** na QAGETI nie zapobiega zatwierdzeniu jednostki pracy.

QAGETA
Operacje pobierania są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora IAIGET z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *InhibitPut (10-cyfrowa liczba całkowita ze znakiem) dla IBM i*

Określa, czy operacje umieszczania dla tej kolejki są dozwolone.

Tabela 778. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jeśli w ścieżce rozstrzygania nazw kolejek znajduje się więcej niż jedna definicja, operacje umieszczania muszą być dozwolone dla *każdej* definicji w ścieżce (w tym dla wszystkich definicji aliasów menedżera kolejek) w czasie operacji umieszczania, aby wywołanie MQPUT lub MQPUT1 mogło zakończyć się powodzeniem. Może mieć jedną z następujących wartości:

QAPUTI
Operacje umieszczania są zablokowane.

Wywołania MQPUT i MQPUT1 kończą się niepowodzeniem z kodem przyczyny RC2051.

Uwaga: Jeśli wywołanie MQPUT działające w ramach jednostki pracy zakończy się pomyślnie, zmiana wartości atrybutu **InhibitPut** na QAPUTI nie uniemożliwi zatwierdzenia jednostki pracy.

QAPUTA
Operacje umieszczania (put) są dozwolone.

Aby określić wartość tego atrybutu, należy użyć selektora IAIPUT z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *InitiationQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki inicjującej.

Tabela 779. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to nazwa kolejki zdefiniowana w menedżerze kolejek lokalnych; kolejka musi być typu QTLOC. Menedżer kolejek wysyła komunikat wyzwalacza do kolejki inicjującej, gdy w wyniku komunikatu docierającego do kolejki, do której należy ten atrybut, wymagane jest uruchomienie aplikacji. Kolejka inicjowania musi być monitorowana przez aplikację monitora wyzwalacza, która uruchomi odpowiednią aplikację po odebraniu komunikatu wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora CAINIQ z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *MaxMsgDługość (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalna długość komunikatu w bajtach.

Tabela 780. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to górny limit długości najdłuższego *fizycznego* komunikatu, który można umieścić w kolejce. Jednak ze względu na to, że atrybut kolejki **MaxMsgLength** można ustawić niezależnie od atrybutu menedżera kolejek systemu **MaxMsgLength** , rzeczywisty górny limit długości najdłuższego komunikatu fizycznego, który można umieścić w kolejce, jest mniejszy z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, aplikacja może umieścić w programie MQMD komunikat *logiczny* , który jest dłuższy niż mniejsza z dwóch atrybutów **MaxMsgLength** , ale tylko wtedy, gdy w aplikacji określono flagę MFSEGA. Jeśli ta opcja jest określona, górny limit długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zazwyczaj ograniczenia zasobów narzucane przez system operacyjny lub środowisko, w którym działa aplikacja, skutkuje niższym limitem.

Próba umieszczenia w kolejce zbyt długiego komunikatu nie powiodła się z kodem przyczyny:

- RC2030 , jeśli komunikat jest zbyt duży dla kolejki
- RC2031 , jeśli komunikat jest zbyt duży dla menedżera kolejek, ale nie zbyt duży dla kolejki

Dolny limit dla atrybutu **MaxMsgLength** wynosi zero. Górny limit jest określany przez środowisko:

- W systemie IBM imaksymalna długość komunikatu wynosi 100 MB (104 857 600 bajtów).

Więcej informacji na ten temat zawiera opis parametru **BUFLEN** w sekcji “MQPUT (umieszczaj komunikat) w systemie IBM i” na stronie 1374.

Aby określić wartość tego atrybutu, należy użyć selektora IAMLEN z wywołaniem MQINQ.

IBM i *MaxQDepth (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalna głębokość kolejki.

Tabela 781. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to zdefiniowany górny limit liczby komunikatów fizycznych, które mogą istnieć w kolejce w dowolnym momencie. Próba umieszczenia komunikatu w kolejce, która zawiera już komunikaty *MaxQDepth* , nie powiodła się z kodem przyczyny RC2053.

Zarówno przetwarzanie jednostki pracy, jak i segmentacja komunikatów mogą spowodować przekroczenie rzeczywistej liczby komunikatów fizycznych w kolejce *MaxQDepth*. Nie ma to jednak wpływu na możliwość pobierania komunikatów- *wszystkie* komunikaty w kolejce mogą być pobierane za pomocą wywołania MQGET w normalny sposób.

Wartość tego atrybutu wynosi zero lub więcej. Górny limit jest określany przez środowisko.

Uwaga: Przestrzeń pamięci dostępna dla kolejki może zostać wyczerpana, nawet jeśli w kolejce znajduje się mniej niż *MaxQDepth* komunikatów.

Aby określić wartość tego atrybutu, należy użyć selektora IAMDEP z wywołaniem MQINQ.

IBM i *MediaLog (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Tożsamość przydziału dziennika (lub dziennika w systemie IBM i) potrzebne do odzyskiwania nośników określonej kolejki.

Tabela 782. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

W menedżerach kolejek, w których używane jest rejestrowanie cykliczne, wartość jest zwracana jako łańcuch o wartości NULL.

IBM i *MsgDeliverySequence (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Kolejność dostarczania komunikatów.

Tabela 783. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Określa to kolejność, w jakiej komunikaty są zwracane do aplikacji przez wywołanie MQGET:

MSFIFO,

Komunikaty są zwracane w kolejności FIFO (pierwszy przyszedł, pierwszy wyszedł).

Oznacza to, że wywołanie MQGET zwróci *pierwszy* komunikat, który spełnia kryteria wyboru określone w wywołaniu, niezależnie od priorytetu komunikatu.

MSPRIO,
Komunikaty są zwracane w kolejności priorytetów.

Oznacza to, że wywołanie MQGET zwróci komunikat *o najwyższym priorytecie* , który spełnia kryteria wyboru określone w wywołaniu. W ramach każdego poziomu priorytetu komunikaty są zwracane w kolejności FIFO (pierwszy przyszedł, pierwszy wyszedł).

Jeśli odpowiednie atrybuty zostaną zmienione, gdy w kolejce znajdują się komunikaty, sekwencja dostarczania będzie następująca:

- Kolejność, w jakiej komunikaty są zwracane przez wywołanie MQGET, jest określana na podstawie wartości atrybutów **MsgDeliverySequence** i **DefPriority** obowiązujących dla kolejki w momencie nadejścia komunikatu do kolejki:
  - Jeśli w momencie nadejścia komunikatu *MsgDeliverySequence* ma wartość MSFIFO, komunikat jest umieszczany w kolejce tak, jakby jego priorytet miał wartość *DefPriority*. Nie ma to wpływu na wartość pola *MDPRI* w deskryptorze komunikatu. To pole zachowuje wartość, którą miało podczas pierwszego umieszczania komunikatu.
  - Jeśli w momencie nadejścia komunikatu *MsgDeliverySequence* ma wartość MSPRIO, komunikat jest umieszczany w kolejce w miejscu odpowiednim dla priorytetu nadanego przez pole *MDPRI* w deskryptorze komunikatu.

  Jeśli wartość atrybutu **MsgDeliverySequence** zostanie zmieniona, gdy w kolejce znajdują się komunikaty, kolejność komunikatów w kolejce nie zostanie zmieniona.

  Jeśli wartość atrybutu **DefPriority** zostanie zmieniona, gdy w kolejce znajdują się komunikaty, komunikaty nie muszą być dostarczane w kolejności FIFO, nawet jeśli atrybut **MsgDeliverySequence** jest ustawiony na MSFIFO; te, które zostały umieszczone w kolejce z wyższym priorytetem, są dostarczane jako pierwsze.

Aby określić wartość tego atrybutu, należy użyć selektora IAMDS z wywołaniem MQINQ.

IBM i *OpenInputLiczba (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Liczba operacji otwierania do wprowadzania.

Tabela 784. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X				

Jest to liczba uchwytów, które są obecnie poprawne na potrzeby usuwania komunikatów z kolejki za pomocą wywołania MQGET. Jest to łączna liczba takich uchwytów znanych *lokalnemu* menedżerowi kolejek. Jeśli kolejka jest kolejką współużytkowaną, licznik nie obejmuje operacji otwierania do wprowadzania, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych.

Liczba ta obejmuje uchwyty, w których kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla danych wejściowych. Licznik nie obejmuje uchwytów, w których otwarto kolejkę dla działań, które nie zawierały danych wejściowych (na przykład kolejka otwarta tylko do przeglądania).

Wartość tego atrybutu zmienia się w miarę działania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IAOIC z wywołaniem MQINQ.

IBM i *OpenOutputLiczebność (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Liczba operacji otwierania dla danych wyjściowych.

Tabela 785. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X				

Jest to liczba uchwytów, które są obecnie poprawne na potrzeby dodawania komunikatów do kolejki za pomocą wywołania MQPUT. Jest to łączna liczba takich uchwytów znanych *lokalnemu* menedżerowi kolejek. Nie obejmuje to operacji otwierania do wyprowadzania wykonanych dla tej kolejki w zdalnych menedżerach kolejek. Jeśli kolejka jest kolejką współużytkowaną, liczba ta nie obejmuje otwartych danych wyjściowych, które zostały wykonane dla kolejki w innych menedżerach kolejek w grupie współużytkowania kolejek, do której należy menedżer kolejek lokalnych.

Liczba ta obejmuje uchwyty, w których kolejka aliasowa, która jest tłumaczona na tę kolejkę, została otwarta dla danych wyjściowych. Liczba ta nie obejmuje uchwytów, w których otwarto kolejkę dla działań, które nie zawierały danych wyjściowych (na przykład kolejka otwarta tylko w celu wykonania zapytania).

Wartość tego atrybutu zmienia się w miarę działania menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IAOOC z wywołaniem MQINQ.

IBM i *ProcessName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa procesu.

Tabela 786. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to nazwa obiektu procesu, który jest zdefiniowany w lokalnym menedżerze kolejek. Obiekt procesu identyfikuje program, który może obsługiwać kolejkę.

Aby określić wartość tego atrybutu, należy użyć selektora CAPRON z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNPRON.

IBM i *Zdarzenie QDepthHigh(10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia nadmiaru kolejki.

Tabela 787. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Zdarzenie Duże zapełnienie kolejki wskazuje, że aplikacja umieściła komunikat w kolejce, co spowodowało, że liczba komunikatów w kolejce stała się większa lub równa wysokiemu progowi zapełnienia kolejki (patrz atrybut **QDepthHighLimit**).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Zdarzenie QDepthHighmoże mieć jedną z dwóch wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDHE z wywołaniem MQINQ.

IBM i *QDepthHighLimit (10-cyfrowa liczba całkowita ze znakiem) on IBM i*

Górny limit głębokości kolejki.

Tabela 788. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to próg, z którym porównywane jest zapełnienie kolejki w celu wygenerowania zdarzenia dużego zapełnienia kolejki. To zdarzenie wskazuje, że aplikacja umieściła komunikat w kolejce, co spowodowało, że liczba komunikatów w kolejce stała się większa lub równa wysokiemu progowi zapełnienia kolejki. Patrz atrybut **QDepthHighEvent** .

Wartość jest wyrażona jako procent maksymalnego zapełnienia kolejki (atrybut**MaxQDepth**) i należy do zakresu od 0 do 100. Wartość domyślna to 80.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDHL z wywołaniem MQINQ.

IBM i *QDepthLowZdarzenie (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia niedoboru kolejki.

Tabela 789. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Zdarzenie niedoboru kolejki wskazuje, że aplikacja pobrała komunikat z kolejki, co spowodowało, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnemu progowi zapełnienia kolejki (patrz atrybut **QDepthLowLimit**).

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Zdarzenie QDepthLowmoże mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDLE z wywołaniem MQINQ.

IBM i *QDepthLowLimit (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Dolny limit głębokości kolejki.

Tabela 790. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to próg, z którym porównywane jest zapełnienie kolejki w celu wygenerowania zdarzenia niedoboru kolejki. To zdarzenie wskazuje, że aplikacja pobrała komunikat z kolejki, co spowodowało, że liczba komunikatów w kolejce stała się mniejsza lub równa dolnemu progowi zapełnienia kolejki. Patrz atrybut **QDepthLowEvent** .

Wartość jest wyrażona jako procent maksymalnego zapełnienia kolejki (atrybut**MaxQDepth**) i należy do zakresu od 0 do 100. Wartością domyślną jest 20.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDLL z wywołaniem MQINQ.

IBM i *QDepthMaxZdarzenie (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia zapełnienia kolejki.

Tabela 791. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Zdarzenie zapełnienia kolejki wskazuje, że operacja umieszczenia w kolejce została odrzucona, ponieważ kolejka jest pełna, czyli zapełnienie kolejki osiągnęło już maksymalną wartość.

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Może mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAQDME z wywołaniem MQINQ.

IBM i *QDesc (64-bitowy łańcuch znaków) w systemie IBM i*

Opis kolejki.

Tabela 792. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X	X	X	X

Jest to pole, którego można użyć jako komentarza opisowego. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole może zawierać znaki DBCS (z maksymalną długością pola wynoszącą 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAQD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQD.

IBM i *QName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki.

Tabela 793. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	X

Jest to nazwa kolejki zdefiniowana w menedżerze kolejek lokalnych. Więcej informacji na temat nazw kolejek zawiera sekcja Reguły nazewnictwa obiektów IBM MQ. Wszystkie kolejki zdefiniowane w menedżerze kolejek współużytkują tę samą przestrzeń nazw kolejki. Dlatego kolejka QTLOC i kolejka QTALS nie mogą mieć takiej samej nazwy.

Aby określić wartość tego atrybutu, należy użyć selektora CAQN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *QServiceInterval (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Cel dla odstępu czasu usługi kolejki.

Tabela 794. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to odstęp czasu usługi używany do porównywania w celu wygenerowania zdarzeń wysokiego i prawidłowego odstępu czasu usługi. Patrz atrybut **`QServiceIntervalEvent`** .

Wartość jest wyrażona w milisekundach i mieści się w zakresie od 0 do 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora IAQSI z wywołaniem MQINQ.

IBM i *QServiceIntervalZdarzenie (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia Wysoki odstęp czasu usługi, czy OK odstępu czasu usługi.

Tabela 795. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

- Zdarzenie wysokiego odstępu czasu usługi jest generowane, gdy sprawdzenie wskazuje, że nie pobrano żadnych komunikatów z kolejki przez co najmniej jeden czas wskazany przez atrybut **`QServiceInterval`** .
- Zdarzenie OK odstępu czasu usługi jest generowane, gdy sprawdzenie wskazuje, że komunikaty zostały pobrane z kolejki w czasie wskazanym przez atrybut **`QServiceInterval`** .

Uwaga: Wartość tego atrybutu może zmieniać się dynamicznie.

Ten atrybut może mieć jedną z następujących wartości:

QSIEJJ
Włączone zdarzenia wysokiego odstępu czasu usługi kolejki.

- Zdarzenia wysokiego odstępu czasu usługi kolejki są **włączone** i
- Zdarzenia OK odstępu czasu usługi kolejki są **wyłączone**.

QSIEOK
Włączono zdarzenia OK odstępu czasu usługi kolejki.

- Zdarzenia wysokiego odstępu czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK odstępu czasu usługi kolejki są **włączone**.

QSIENO
Nie włączono żadnych zdarzeń odstępu czasu usługi kolejki.

- Zdarzenia wysokiego odstępu czasu usługi kolejki są **wyłączone** i
- Zdarzenia OK odstępu czasu usługi kolejki są również **wyłączone**.

W przypadku kolejek współużytkowanych wartość tego atrybutu jest ignorowana; przyjmowana jest wartość QSIENO.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAQSIE z wywołaniem MQINQ.

IBM i *QSGDisp (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Dyspozycja grupy współużytkowania kolejek.

Tabela 796. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	

Określa dyspozycję kolejki. Jest to jedna z następujących wartości:

QSGDQM,
Dyspozycja menedżera kolejek.

Obiekt ma dyspozycję menedżera kolejek. Oznacza to, że definicja obiektu jest znana tylko menedżerowi kolejek lokalnych; definicja nie jest znana innym menedżerom kolejek w grupie współużytkowania kolejek.

Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć obiekt o takiej samej nazwie i typie jak bieżący obiekt, ale są to oddzielne obiekty i nie ma między nimi korelacji. Ich atrybuty nie są ograniczone do siebie nawzajem.

QSGDCP,
Dyspozycja skopiowanego obiektu.

Obiekt jest lokalną kopią definicji obiektu głównego, która istnieje we współużytkowanym repozytorium. Każdy menedżer kolejek w grupie współużytkowania kolejek może mieć własną kopię obiektu. Początkowo wszystkie kopie mają takie same atrybuty, ale za pomocą komend MQSC każda kopia może zostać zmieniona w taki sposób, że jej atrybuty różnią się od atrybutów innych kopii. Atrybuty kopii są ponownie synchronizowane, gdy definicja główna w repozytorium współużytkowanym zostanie zmieniona.

QSGDSH,
Dyspozycja współużytkowana.

Obiekt ma dyspozycję współużytkowaną. Oznacza to, że we współużytkowanym repozytorium istnieje pojedyncza instancja obiektu, która jest znana wszystkim menedżerom kolejek w grupie współużytkowania kolejek. Gdy menedżer kolejek w grupie uzyskuje dostęp do obiektu, uzyskuje dostęp do pojedynczej współużytkowanej instancji obiektu.

Aby określić wartość tego atrybutu, należy użyć selektora IAQSGD z wywołaniem MQINQ.

z/OS Ten atrybut jest obsługiwany tylko w systemie z/OS.

IBM i *QType (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Typ kolejki.

Tabela 797. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X		X	X	X

Ten atrybut ma jedną z następujących wartości:

QTALS
Definicja kolejki aliasowej.

QTCLUS
Kolejka klastra.

QTLOC,
Kolejka lokalna.

QTREM
Lokalna definicja kolejki zdalnej.

Aby określić wartość tego atrybutu, należy użyć selektora IAQTYP z wywołaniem MQINQ.

IBM i *RemoteQMgrNazwa (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa zdalnego menedżera kolejek.

Tabela 798. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
			X	

Jest to nazwa zdalnego menedżera kolejek, w którym zdefiniowano kolejkę *RemoteQName* . Jeśli kolejka *RemoteQName* ma wartość *QSGDisp* QSGDCP lub QSGDSH, *RemoteQMgrName* może być nazwą grupy współużytkowania kolejek, która jest właścicielem *RemoteQName*.

Jeśli aplikacja otwiera lokalną definicję kolejki zdalnej, pole *RemoteQMgrName* nie może być puste i nie może być nazwą lokalnego menedżera kolejek. Jeśli parametr *XmitQName* jest pusty, jako kolejka transmisji używana jest kolejka lokalna o takiej samej nazwie jak *RemoteQMgrName* . Jeśli nie ma kolejki o nazwie *RemoteQMgrName*, używana jest kolejka identyfikowana przez atrybut **DefXmitQName** menedżera kolejek.

Jeśli ta definicja jest używana dla aliasu menedżera kolejek, *RemoteQMgrName* jest nazwą menedżera kolejek, który jest aliasowany. Może to być nazwa lokalnego menedżera kolejek. W przeciwnym razie, jeśli parametr *XmitQName* jest pusty podczas otwierania, musi istnieć kolejka lokalna o takiej samej nazwie jak *RemoteQMgrName* ; Ta kolejka jest używana jako kolejka transmisji.

Jeśli ta definicja jest używana dla aliasu odpowiedzi, ta nazwa jest nazwą menedżera kolejek, który ma być *MDRM*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora CARQMN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQMN.

IBM i *RemoteQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki zdalnej.

Tabela 799. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
			X	

Jest to nazwa kolejki znana w zdalnym menedżerze kolejek *RemoteQMgrName*.

Jeśli aplikacja otwiera lokalną definicję kolejki zdalnej, w momencie otwarcia *RemoteQName* nie może być pusta.

Jeśli ta definicja jest używana dla definicji aliasu menedżera kolejek, po otwarciu *RemoteQName* musi być pusta.

Jeśli definicja jest używana dla aliasu odpowiedzi, ta nazwa jest nazwą kolejki, która ma być nazwą *MDRQ*.

Uwaga: Podczas tworzenia lub modyfikowania definicji kolejki nie jest wykonywane sprawdzanie poprawności wartości określonej dla tego atrybutu.

Aby określić wartość tego atrybutu, należy użyć selektora CARQN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *RetentionInterval (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Interwał czasu przechowywania.

Tabela 800. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Jest to czas, w którym kolejka powinna zostać zachowana. Po upływie tego czasu kolejka zostanie zakwalifikowana do usunięcia.

Czas jest mierzony w godzinach, licząc od daty i godziny utworzenia kolejki. Data utworzenia kolejki jest rejestrowana w pliku *CreationDate* , a godzina utworzenia kolejki jest rejestrowana w atrybucie **CreationTime** .

Te informacje umożliwiają aplikacji konserwacyjnej lub operatorowi identyfikowanie i usuwanie kolejek, które nie są już wymagane.

Uwaga: Menedżer kolejek nigdy nie próbuje usunąć kolejek na podstawie tego atrybutu lub zapobiec usuwaniu kolejek z okresem przechowywania, który nie utracił ważności. To użytkownik jest odpowiedzialny za wykonanie wszystkich wymaganych działań.

Aby zapobiec gromadzeniu trwałych kolejek dynamicznych, należy użyć realistycznego odstępu czasu przechowywania (patrz sekcja *DefinitionType*). Atrybut ten może być jednak również używany z predefiniowanymi kolejkami.

Aby określić wartość tego atrybutu, należy użyć selektora IARINT z wywołaniem MQINQ.

IBM i *Zasięg (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy pozycja dla tej kolejki istnieje również w katalogu komórki.

Tabela 801. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X		X	X	

Katalog komórki jest udostępniany przez instalowalną usługę nazw. Może mieć jedną z następujących wartości:

WYNIK
Zasięg menedżera kolejek.

Definicja kolejki ma zasięg menedżera kolejek. Oznacza to, że definicja kolejki nie wykracza poza menedżera kolejek, który jest jej właścicielem. Aby otworzyć kolejkę dla danych wyjściowych z innego menedżera kolejek, należy określić nazwę menedżera kolejek będącego właścicielem lub inny menedżer kolejek musi mieć lokalną definicję kolejki.

ZAKRES
Zasięg komórki.

Definicja kolejki ma zasięg komórki. Oznacza to, że definicja kolejki jest również umieszczana w katalogu komórki dostępnym dla wszystkich menedżerów kolejek w komórce. Kolejkę można otworzyć do wyjścia z dowolnego menedżera kolejek w komórce, podając tylko nazwę kolejki. Nazwa menedżera kolejek, do którego należy kolejka, nie musi być określona. Jednak definicja kolejki nie jest dostępna dla żadnego menedżera kolejek w komórce, który również ma lokalną definicję kolejki o tej nazwie, ponieważ definicja lokalna ma pierwszeństwo.

Katalog komórek jest udostępniany przez instalowalną usługę nazw, taką jak LDAP (Lightweight Directory Access Protocol). Należy zauważyć, że system IBM MQ nie obsługuje już usługi nazw DCE (Distributed Computing Environment), która była wcześniej używana do wstawiania definicji kolejek do katalogu DCE (również nie jest już obsługiwana).

Kolejki modelowe i dynamiczne nie mogą mieć zasięgu komórki.

Ta wartość jest poprawna tylko wtedy, gdy skonfigurowano usługę nazw obsługującą katalog komórki.

Aby określić wartość tego atrybutu, należy użyć selektora IASCOP z wywołaniem MQINQ.

Obsługa tego atrybutu podlega następującym ograniczeniom:

- W systemie IBM iatrybut jest obsługiwany, ale poprawny jest tylko atrybut SCOQM.

IBM i *Współużytkowalność (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy kolejka może być współużytkowana dla danych wejściowych.

Tabela 802. Typy kolejek, do których ma zastosowanie ten atrybut				
Lokalna	**Model**	**Alias**	**Zdalna**	**Klaster**
X	X			

Wskazuje, czy kolejka może być otwierana na potrzeby wielokrotnego współbieżnego wprowadzania. Może mieć jedną z następujących wartości:

QASHR (Kadry)
Kolejka jest współużytkowana.

Dozwolone jest wielokrotne otwieranie z opcją OOINPS.

KANSHR
Nie można współużytkować kolejki.

Wywołanie MQOPEN z opcją OOINPS jest traktowane jako OOINPX.

Aby określić wartość tego atrybutu, należy użyć selektora IASHAR z wywołaniem MQINQ.

IBM i *TriggerControl (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Element sterujący wyzwalacza.

Tabela 803. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Określa, czy komunikaty wyzwalacza są zapisywane w kolejce inicjującej w celu uruchomienia aplikacji do obsługi kolejki. Jest to jeden z następujących elementów:

TCOFF,
Komunikaty wyzwalacza nie są wymagane.

Dla tej kolejki nie będą zapisywane żadne komunikaty wyzwalacza. W tym przypadku wartość *TriggerType* jest nieistotna.

TCON (ikona)
Wymagane są komunikaty wyzwalacza.

Komunikaty wyzwalacza mają być zapisywane dla tej kolejki, gdy wystąpią odpowiednie zdarzenia wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGC z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *TriggerData (64-bitowy łańcuch znaków) w systemie IBM i*

Dane wyzwalacza.

Tabela 804. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Są to dane w formacie wolnym, które menedżer kolejek wstawia do komunikatu wyzwalacza, gdy komunikat pojawiający się w tej kolejce powoduje zapisanie komunikatu wyzwalacza w kolejce inicjującej.

Treść tych danych nie ma znaczenia dla menedżera kolejek. Ma znaczenie albo dla aplikacji monitorującej wyzwalacz, która przetwarza kolejkę inicjującą, albo dla aplikacji uruchamianej przez monitor wyzwalacza.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora CATRGD z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET. Długość tego atrybutu jest określona przez LNTRGD.

IBM i *TriggerDepth (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Wyzwalacz uruchamiany zapełnieniem.

Tabela 805. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to liczba komunikatów o priorytecie *TriggerMsgPriority* lub większym, które muszą znajdować się w kolejce przed zapisaniem komunikatu wyzwalacza. Ma to zastosowanie, gdy parametr *TriggerType* jest ustawiony na wartość TTDPTH. Wartość parametru *TriggerDepth* wynosi jeden lub więcej. W przeciwnym razie ten atrybut nie jest używany.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGD z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *TriggerMsgPriorytet (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Priorytet komunikatu progu dla wyzwalaczy w systemie IBM MQ for IBM i.

Tabela 806. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Jest to priorytet komunikatu, poniżej którego komunikaty nie przyczyniają się do generowania komunikatów wyzwalacza (oznacza to, że menedżer kolejek ignoruje te komunikaty podczas określania, czy komunikat wyzwalacza powinien zostać wygenerowany). Wartość *TriggerMsgPriority* może być z zakresu od 0 (najniższy) do *MaxPriority* (najwyższy; patrz "Atrybuty menedżera kolejek w systemie IBM i" na stronie 1443). Wartość zero powoduje, że wszystkie komunikaty mają wpływ na generowanie komunikatów wyzwalacza.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGP z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *TriggerType (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Typ wyzwalacza.

Tabela 807. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Steruje to warunkami, w jakich komunikaty wyzwalacza są zapisywane w wyniku komunikatów przychodzących do tej kolejki. Jest to jedna z następujących wartości:

TTTNONE
Brak komunikatów wyzwalacza.

W wyniku komunikatów w tej kolejce nie są zapisywane żadne komunikaty wyzwalacza. Ma to taki sam efekt, jak ustawienie parametru *TriggerControl* na wartość TCOFF.

TFRST
Wyzwalaj komunikat, gdy zapełnienie kolejki jest z zakresu od 0 do 1.

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce zmieni się z 0 na 1.

TTEVRY
Komunikat wyzwalacza dla każdego komunikatu.

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy w kolejce pojawi się komunikat o priorytecie *TriggerMsgPriority* lub większym.

TTTDPTH
Komunikat wyzwalacza po przekroczeniu progu głębokości.

Komunikat wyzwalacza jest zapisywany za każdym razem, gdy liczba komunikatów o priorytecie *TriggerMsgPriority* lub większej w kolejce jest równa lub większa niż *TriggerDepth*. Po zapisaniu komunikatu wyzwalacza *TriggerControl* jest ustawiany na wartość TCOFF, aby zapobiec dalszemu wyzwalaniu, dopóki nie zostanie jawnie włączony.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGT z wywołaniem MQINQ. Aby zmienić wartość tego atrybutu, należy użyć wywołania MQSET.

IBM i *Użycie (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Użycie kolejki.

Tabela 808. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
X	X			

Wskazuje, do czego jest używana kolejka. Jest to jedna z następujących wartości:

USNORMA
Normalne użycie.

Jest to kolejka używana przez normalne aplikacje podczas umieszczania i pobierania komunikatów. Kolejka nie jest kolejką transmisji.

USTRAN
Kolejka transmisji.

Jest to kolejka używana do przechowywania komunikatów przeznaczonych dla zdalnych menedżerów kolejek. Gdy normalna aplikacja wysyła komunikat do kolejki zdalnej, menedżer kolejek lokalnych przechowuje komunikat tymczasowo w odpowiedniej kolejce transmisji w specjalnym formacie. Następnie agent kanału komunikatów odczytuje komunikat z kolejki transmisji i transportuje go do zdalnego menedżera kolejek. Więcej informacji na temat kolejek transmisji zawiera sekcja Kolejki transmisji.

Tylko aplikacje uprzywilejowane mogą otwierać kolejkę transmisji dla OOOUT w celu bezpośredniego umieszczania w niej komunikatów. Zwykle oczekuje się, że będą to robić tylko aplikacje narzędziowe. Należy uważać, aby format danych komunikatu był poprawny (patrz sekcja “MQXQH (nagłówek kolejki transmisji) w systemie IBM i” na stronie 1284). W przeciwnym razie podczas procesu transmisji mogą wystąpić błędy. Kontekst nie jest przekazywany ani ustawiany, chyba że określono jedną z opcji kontekstu PM*.

Aby określić wartość tego atrybutu, należy użyć selektora IAUSAG z wywołaniem MQINQ.

IBM i *XmitQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki transmisji.

Tabela 809. Typy kolejek, do których ma zastosowanie ten atrybut

Lokalna	Model	Alias	Zdalna	Klaster
			X	

Jeśli ten atrybut nie jest pusty podczas otwierania, dla kolejki zdalnej lub dla definicji aliasu menedżera kolejek, określa nazwę lokalnej kolejki transmisji, która ma być używana do przekazywania komunikatu.

Jeśli parametr *XmitQName* jest pusty, jako kolejka transmisji używana jest kolejka lokalna o nazwie takiej samej jak *RemoteQMgrName* . Jeśli nie ma kolejki o nazwie *RemoteQMgrName*, używana jest kolejka identyfikowana przez atrybut **DefXmitQName** menedżera kolejek.

Ten atrybut jest ignorowany, jeśli definicja jest używana jako alias menedżera kolejek, a *RemoteQMgrName* jest nazwą lokalnego menedżera kolejek. Atrybut nie jest również brany pod uwagę, jeśli definicja jest używana jako definicja aliasu kolejki zwrotnej.

Aby określić wartość tego atrybutu, należy użyć selektora CAXQN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

Atrybuty listy nazw

W tej sekcji przedstawiono podsumowanie atrybutów specyficznych dla list nazw. Atrybuty są opisane w porządku alfabetycznym.

Uwaga: Wyświetlane nazwy atrybutów są nazwami używanymi w wywołaniach MQINQ i MQSET.

Opisy atrybutów

Obiekt listy nazw ma następujące atrybuty:

AlterationDate (12-bajtowy łańcuch znaków)

Data ostatniej zmiany definicji.

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNDATE.

AlterationTime (8-bajtowy łańcuch znaków)

Czas ostatniej zmiany definicji.

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNTIME.

NameCount (10-cyfrowa liczba całkowita ze znakiem)

Liczba nazw na liście nazw.

Wartość ta jest większa lub równa zero. Zdefiniowana jest następująca wartość:

NCMXNL
Maksymalna liczba nazw na liście nazw.

Aby określić wartość tego atrybutu, należy użyć selektora IANAMC z wywołaniem MQINQ.

NamelistDesc (64-bitowy łańcuch znaków)

Opis listy nazw.

Jest to pole, które może być używane na potrzeby komentarzy opisowych; jego wartość jest określana przez proces definiowania. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole to może zawierać znaki DBCS (maksymalna długość pola wynosi 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CALSTD z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez LNNLD.

NamelistName (48-bajtowy łańcuch znaków)

Nazwa listy nazw.

Jest to nazwa listy nazw, która jest zdefiniowana w menedżerze kolejek lokalnych.

Każda lista nazw ma nazwę inną niż nazwy innych list nazw należących do menedżera kolejek, ale może duplikować nazwy innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora CALSTN z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez numer LNNLN.

Nazwy (48-bajtowy łańcuch znaków x NameCount)

Lista nazw *NameCount* .

Każda nazwa jest nazwą obiektu, który jest zdefiniowany dla lokalnego menedżera kolejek. Więcej informacji na temat nazw obiektów zawiera sekcja Nazywanie obiektów IBM MQ.

Aby określić wartość tego atrybutu, należy użyć selektora CANAMS z wywołaniem MQINQ.

Długość każdej nazwy na liście jest określona przez LNOBJN.

IBM i Atrybuty definicji procesów w systemie IBM i

W tej sekcji przedstawiono podsumowanie atrybutów specyficznych dla definicji procesów. Atrybuty są opisane w porządku alfabetycznym.

Uwaga: Wyświetlane nazwy atrybutów są nazwami używanymi w wywołaniach MQINQ i MQSET. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Szczegółowe informacje na ten temat zawiera sekcja Komendy MQSC .

Opisy atrybutów

Obiekt definicji procesu ma następujące atrybuty:

AlterationDate (12-bajtowy łańcuch znaków)

Data ostatniej zmiany definicji.

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNDATE.

AlterationTime (8-bajtowy łańcuch znaków)

Czas ostatniej zmiany definicji.

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNTIME.

ApplId (256-bajtowy łańcuch znaków)

Identyfikator aplikacji.

Jest to łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona. Te informacje są używane przez aplikację monitorującą wyzwalacz, która przetwarza komunikaty w kolejce inicjującej. Informacje te są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalającego.

Znaczenie parametru *ApplId* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniony przez IBM MQ wymaga, aby *ApplId* była nazwą programu wykonywalnego.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora CAAPPI z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNPROA.

ApplType (10-cyfrowa liczba całkowita ze znakiem)

Typ aplikacji.

Identyfikuje on rodzaj programu, który ma zostać uruchomiony w odpowiedzi na odebranie komunikatu wyzwalacza. Te informacje są używane przez aplikację monitorującą wyzwalacz, która przetwarza komunikaty w kolejce inicjującej. Informacje te są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalającego.

ApplType może mieć dowolną wartość. Dla typów standardowych można użyć następujących wartości; typy aplikacji zdefiniowane przez użytkownika są ograniczone do wartości z zakresu od ATUFST do ATULST:

oCICS
CICS .

AT400
Aplikacja IBM i .

ATUFST
Najniższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

ATULST
Najwyższa wartość dla typu aplikacji zdefiniowanego przez użytkownika.

Aby określić wartość tego atrybutu, należy użyć selektora IAAPPT z wywołaniem MQINQ.

EnvData (128-bajtowy łańcuch znaków)

Dane środowiska.

Jest to łańcuch znaków zawierający informacje związane ze środowiskiem dotyczące uruchamianej aplikacji. Te informacje są używane przez aplikację monitorującą wyzwalacz, która przetwarza komunikaty w kolejce inicjującej. Informacje te są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalającego.

Znaczenie parametru *EnvData* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniany przez IBM MQ dopisuje *EnvData* do listy parametrów przekazanej do uruchomionej aplikacji. Lista parametrów składa się ze struktury MQTMC2 , po której następuje jedno puste miejsce, po którym następuje łańcuch *EnvData* z usuniętymi odstępami końcowymi.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora CAENVD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNPROE.

ProcessDesc (64-bitowy łańcuch znaków)

Opis procesu.

Jest to pole, którego można użyć jako komentarza opisowego. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole może zawierać znaki DBCS (z maksymalną długością pola wynoszącą 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAPROD z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez LNPROD.

ProcessName (48-bajtowy łańcuch znaków)

Nazwa procesu.

Jest to nazwa definicji procesu, która jest zdefiniowana w menedżerze kolejek lokalnych.

Każda definicja procesu ma nazwę inną niż nazwy innych definicji procesów należących do menedżera kolejek. Jednak nazwa definicji procesu może być taka sama jak nazwa innych obiektów menedżera kolejek różnych typów (na przykład kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora CAPRON z wywołaniem MQINQ.

Długość tego atrybutu jest określona przez LNPRON.

UserData (128-bajtowy łańcuch znaków)

Dane użytkownika.

Jest to łańcuch znaków zawierający informacje o użytkowniku dotyczące uruchamianej aplikacji. Te informacje są przeznaczone dla aplikacji monitora wyzwalacza, która przetwarza komunikaty w kolejce inicjującej lub aplikacji, która jest uruchamiana przez monitor wyzwalacza. Informacje są wysyłane do kolejki inicjującej w ramach komunikatu wyzwalacza.

Znaczenie parametru *UserData* jest określane przez aplikację monitorującą wyzwalacz. Monitor wyzwalacza udostępniony przez IBM MQ przekazuje *UserData* do uruchomionej aplikacji jako część listy parametrów. Lista parametrów składa się ze struktury MQTMC2 (zawierającej łańcuch *UserData*), po której następuje jedno puste miejsce, po którym następuje łańcuch *EnvData* z usuniętymi odstępami końcowymi.

Łańcuch znaków nie może zawierać żadnych wartości pustych. W razie potrzeby jest on dopełniany po prawej stronie odstępami.

Aby określić wartość tego atrybutu, należy użyć selektora CAUSRD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNPROU.

IBM i Atrybuty menedżera kolejek w systemie IBM i

Podsumowanie atrybutów menedżera kolejek.

Niektóre atrybuty menedżera kolejek są stałe dla konkretnych implementacji, a inne można zmienić za pomocą komendy MQSC `ALTER QMGR`. Atrybuty można również wyświetlić za pomocą komendy `DISPLAY QMGR`. Większość atrybutów menedżera kolejek można uzyskać, otwierając specjalny obiekt OTQM i używając wywołania MQINQ ze zwróconym uchwytem.

W poniższej tabeli znajduje się podsumowanie atrybutów specyficznych dla menedżera kolejek. Atrybuty są opisane w porządku alfabetycznym.

Uwaga: Nazwy atrybutów przedstawione w tej sekcji są nazwami używanymi w wywołaniach MQINQ i MQSET. Jeśli do definiowania, modyfikowania lub wyświetlania atrybutów używane są komendy MQSC, używane są alternatywne nazwy skrócone. Więcej informacji na ten temat zawiera sekcja Komendy MQSC .

Tabela 810. Atrybuty menedżera kolejek

Atrybut	Opis
AlterationDate	Data ostatniej zmiany definicji
AlterationTime	Czas ostatniej zmiany definicji
AuthorityEvent	Określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane)
BridgeEvent	Określa, czy generowane są zdarzenia mostu IMS
ChannelAutoOdw	Określa, czy dozwolona jest automatyczna definicja kanału
ChannelAutoDefEvent	Określa, czy generowane są zdarzenia automatycznej definicji kanału
ChannelAutoDefExit	Nazwa procedury zewnętrznej dla automatycznej definicji kanału
ChannelEvent	Określa, czy są generowane zdarzenia kanału

Tabela 810. Atrybuty menedżera kolejek (kontynuacja)	
Atrybut	**Opis**
ClusterCacheTyp	Określa, czy wielkość pamięci podręcznej klastra jest stała, czy też ma być określana dynamicznie
ClusterWorkloadDane	Dane użytkownika dla wyjścia obciążenia klastra
ClusterWorkloadWyjdź	Nazwa programu zewnętrznego dla zarządzania obciążeniem klastra
ClusterWorkloadDługość	Maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra
CodedCharSetId	Identyfikator kodowanego zestawu znaków
CommandEvent	Określa, czy komunikaty zdarzeń komend są umieszczane w kolejce
CommandInputQName	Nazwa kolejki wejściowej komendy
CommandLevel	Poziom komendy
ConfigurationEvent	zdarzenie konfiguracji
DeadLetterQName	Nazwa kolejki niedostarczonych komunikatów
DefClusterXmitQueueTyp	Domyślny typ kolejki transmisji klastra
DefXmitQName	Domyślna nazwa kolejki transmisji
DistLists	Obsługa listy dystrybucyjnej
InhibitEvent	Określa, czy są generowane zdarzenia blokady (zablokuj pobieranie i zablokuj umieszczanie)
LocalEvent	Określa, czy generowane są lokalne zdarzenia błędów
LoggerEvent	Określa, czy generowane są zdarzenia dziennika odtwarzania
MaxHandles	Maksymalna liczba uchwytów
MaxMsgdługość komunikatu	Maksymalna długość komunikatu w bajtach
MaxPriority	Maksymalny priorytet
MaxUncommittedKomunikaty	Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy
PerformanceEvent	Określa, czy generowane są zdarzenia związane z wydajnością
Platforma	Platforma, na której jest uruchomiony menedżer kolejek
TrybPubSub	Określa, czy mechanizm publikowania/subskrypcji i umieszczony w kolejce interfejs publikowania/subskrypcji są uruchomione.
QMgrDesc	Opis menedżera kolejek
QMgrIdentifier	Unikalny wewnętrznie wygenerowany identyfikator menedżera kolejek
QMgrName	Nazwa menedżera kolejek
RemoteEvent	Określa, czy generowane są zdalne zdarzenia błędów
RepositoryName	Nazwa klastra, dla którego ten menedżer kolejek udostępnia usługi repozytorium
RepositoryNamelist	Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługi repozytorium

Tabela 810. Atrybuty menedżera kolejek (kontynuacja)	
Atrybut	**Opis**
Lista nazw SSLCRL	Nazwa obiektu listy nazw zawierającego nazwy obiektów informacji uwierzytelniającej (patrz uwaga 1)
odpowietrznik (SSLEvent)	Określa, czy są generowane zdarzenia TLS
SSLKeyRepository	Położenie repozytorium kluczy TLS (patrz uwaga 1)
Liczba:SSLKeyReset	Określa liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji TLS przed ponownym wynegocjowaniem klucza szyfrowania
ZdarzenieStartStop	Określa, czy generowane są zdarzenia uruchomienia i zatrzymania
SyncPoint	Dostępność punktu synchronizacji
RejestrowanieTraceRoute	Steruje rejestrowaniem informacji o trasie śledzenia dla komunikatów
TreeLifeCzas	Czas życia (w sekundach) tematów nieadministracyjnych
TriggerInterval	Odstęp czasu między komunikatami wyzwalacza

Uwagi:

1. Ten atrybut nie może zostać sprawdzony przy użyciu wywołania MQINQ i nie jest opisany w tej sekcji. Więcej informacji na temat tego atrybutu zawiera sekcja Zmiana menedżera kolejek.

IBM i *AlterationDate (12-bajtowy łańcuch znaków) w systemie IBM i*

Data ostatniej zmiany definicji.

Jest to data ostatniej zmiany definicji. Data ma format YYYY-MM-DDi jest dopełniana dwoma odstępami końcowymi w celu uzyskania długości 12 bajtów.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNDATE.

IBM i *AlterationTime (8-bajtowy łańcuch znaków) w systemie IBM i*

Czas ostatniej zmiany definicji.

Jest to czas ostatniej zmiany definicji. Format godziny to HH.MM.SS.

Aby określić wartość tego atrybutu, należy użyć selektora CAALTT z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNTIME.

IBM i *AuthorityEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia autoryzacji (nieautoryzowane).

Atrybut AuthorityEvent musi mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAAUTE z wywołaniem MQINQ.

IBM i *BridgeEvent (łańcuch znaków) w systemie IBM i*

Ten atrybut określa, czy komunikaty zdarzeń mostu IMS są umieszczane w systemie SYSTEM.ADMIN.CHANNEL.EVENT . Jest on obsługiwany tylko w systemie z/OS.

IBM i *ChannelAutoDef (10-cyfrowa liczba całkowita ze znakiem) on IBM i*

Określa, czy dozwolona jest automatyczna definicja kanału.

Ten atrybut steruje automatyczną definicją kanałów typu CTRCVR i CTSVCN. Należy zauważyć, że automatyczna definicja kanałów CTCLSD jest zawsze włączona. Może mieć jedną z następujących wartości:

CHADDI
Automatyczne definiowanie kanału jest wyłączone.

CHADEN
Automatyczne definiowanie kanału jest włączone.

Aby określić wartość tego atrybutu, należy użyć selektora IACAD z wywołaniem MQINQ.

IBM i *ChannelAutoDefEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia automatycznej definicji kanału.

Dotyczy to kanałów typu CTRCVR, CTSVCN i CTCLSD. Może mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie i wydajność.

Aby określić wartość tego atrybutu, należy użyć selektora IACADE z wywołaniem MQINQ.

IBM i *ChannelAutoDefExit (20-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa procedury zewnętrznej dla automatycznej definicji kanału.

Jeśli ta nazwa nie jest pusta, a parametr *ChannelAutoDef* ma wartość CHADEN, wyjście jest wywoływane za każdym razem, gdy menedżer kolejek ma utworzyć definicję kanału. Dotyczy to kanałów typu CTRCVR, CTSVCN i CTCLSD. Wyjście może następnie wykonać jedną z następujących czynności:

- Zezwalaj na kontynuowanie tworzenia definicji kanału bez wprowadzania zmian.
- Zmodyfikuj atrybuty tworzonej definicji kanału.
- Całkowicie pomijaj tworzenie kanału.

Aby określić wartość tego atrybutu, należy użyć selektora CACADX z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNEXN.

IBM i *ChannelEvent (łańcuch znaków) w systemie IBM i*

Określa, czy generowane są komunikaty o zdarzeniach kanału.

Ten atrybut określa, czy komunikaty zdarzeń kanału są umieszczane w systemie SYSTEM.ADMIN.CHANNEL.EVENT queue, a jeśli tak, to jaki typ komunikatów jest umieszczany w kolejce (na przykład 'channel started', 'channel stopped', 'channel not activated'). Przed implementacją tego atrybutu jedynym sposobem na uniemożliwienie kolejkowania komunikatów zdarzeń kanału było usunięcie kolejki docelowej.

Ten atrybut umożliwia również gromadzenie tylko zdarzeń mostu IMS (ponieważ można teraz wyłączyć zdarzenia kanału, nie są one umieszczane w tej samej kolejce). To samo dotyczy zdarzeń TLS, które mogą być również gromadzone bez konieczności gromadzenia zdarzeń kanału.

Ten atrybut umożliwia również gromadzenie tylko ważnych zdarzeń (na przykład, gdy w kanałach występują błędy, a nie gdy są one uruchamiane i zatrzymywane normalnie).

Atrybut ChannelEvent może mieć jedną z następujących wartości:

- EVREXP (generowane są tylko następujące zdarzenia kanału: RC2279, RC2283, RC2284, RC2295, RC2296).
- EVRENA (generowane są wszystkie zdarzenia kanału; oprócz zdarzeń wygenerowanych przez EVREXP generowane są także zdarzenia RC2282i RC2283).
- EVRDIS (nie są generowane żadne zdarzenia kanału; jest to początkowa wartość domyślna menedżera kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora IACHNE z wywołaniem MQINQ.

IBM i *ClusterCacheTyp (32-bajtowy łańcuch znaków) w systemie IBM i*

Określa, czy wielkość pamięci podręcznej klastra jest stała, czy też dynamiczna.

Jest to zdefiniowany przez użytkownika 32-bajtowy łańcuch znaków, który jest przekazywany do wyjścia obciążenia klastra w momencie jego wywołania. Jeśli nie ma danych do przekazania do wyjścia, łańcuch jest pusty.

Aby określić wartość tego atrybutu, należy użyć selektora CACLWD z wywołaniem MQINQ.

IBM i *ClusterWorkloadData (32-bajtowy łańcuch znaków) w systemie IBM i*

Dane użytkownika dla wyjścia obciążenia klastra.

Jest to zdefiniowany przez użytkownika 32-bajtowy łańcuch znaków, który jest przekazywany do wyjścia obciążenia klastra w momencie jego wywołania. Jeśli nie ma danych do przekazania do wyjścia, łańcuch jest pusty.

Aby określić wartość tego atrybutu, należy użyć selektora CACLWD z wywołaniem MQINQ.

IBM i *ClusterWorkloadExit (20-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa programu zewnętrznego dla zarządzania obciążeniem klastra.

Jeśli ta nazwa nie jest pusta, wyjście jest wywoływane za każdym razem, gdy komunikat jest umieszczany w kolejce klastra lub przenoszony z jednej kolejki nadawczej klastra do innej. Wyjście może następnie zaakceptować instancję kolejki wybraną przez menedżera kolejek jako miejsce docelowe komunikatu lub wybrać inną instancję kolejki.

Aby określić wartość tego atrybutu, należy użyć selektora CACLWX z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNEXN.

IBM i *ClusterWorkloadDługość (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra.

Jest to maksymalna długość danych komunikatu przekazywanych do wyjścia obciążenia klastra. Rzeczywista długość danych przekazywanych do wyjścia wynosi co najmniej:

- Długość komunikatu.
- Atrybut **MaxMsgLength** menedżera kolejek.
- Atrybut **ClusterWorkloadLength** .

Aby określić wartość tego atrybutu, należy użyć selektora IACLWL z wywołaniem MQINQ.

IBM i *CodedCharSetId (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Identyfikator kodowanego zestawu znaków.

Definiuje zestaw znaków używany przez menedżera kolejek dla wszystkich pól łańcucha znaków zdefiniowanych w MQI, takich jak nazwy obiektów oraz data i godzina utworzenia kolejki. Zestaw znaków musi być taki, który zawiera znaki jednobajtowe dla znaków, które są poprawne w nazwach obiektów. Nie ma ono zastosowania do danych aplikacji zawartych w komunikacie. Wartość zależy od środowiska:

- W systemie IBM iwartością jest wartość ustawiona w środowisku, gdy menedżer kolejek jest tworzony po raz pierwszy.

Aby określić wartość tego atrybutu, należy użyć selektora IACCSI z wywołaniem MQINQ.

IBM i *CommandEvent (liczba całkowita) w systemie IBM i*

Określa, czy komunikaty są umieszczane w kolejce lokalnej podczas wydawania komend.

Określa, czy komunikaty są zapisywane w nowej kolejce zdarzeń SYSTEM.ADMIN.COMMAND.EVENT, za każdym razem, gdy wydawane są komendy. Ta funkcja jest użyteczna w przypadku powiadamiania o śledzeniu komend i diagnozowania problemów. Aby uzyskać informacje na temat atrybutu menedżera kolejek CommandEvent , należy użyć nowego selektora atrybutów iacev z jedną z następujących wartości:

- EVRENA-komunikaty zdarzeń komend są generowane i umieszczane w kolejce dla wszystkich pomyślnych komend.
- Komunikaty zdarzeń komend EVND są generowane i umieszczane w kolejce dla wszystkich pomyślnych komend innych niż DISPLAY (MQSC) i Inquire (PCF).
- EVRDIS-komunikaty zdarzeń komend nie są generowane ani umieszczane w kolejce (jest to początkowa wartość domyślna menedżera kolejek).

Aby określić wartość tego atrybutu, należy użyć selektora CMDEV z wywołaniem MQINQ.

IBM i *CommandInputQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki wejściowej komend.

CommandInputNazwa QName jest nazwą kolejki wejściowej komend zdefiniowaną w lokalnym menedżerze kolejek. Jest to kolejka, do której użytkownicy mogą wysyłać komendy, jeśli są do tego uprawnieni. Nazwa kolejki zależy od środowiska:

- W systemie IBM inazwa kolejki to SYSTEM.ADMIN.COMMAND.QUEUEi można do niej wysyłać tylko komendy PCF. Jednak komenda MQSC może zostać wysłana do tej kolejki, jeśli komenda MQSC jest ujęta w komendzie PCF typu CMESC. Więcej informacji na temat komendy Escape zawiera sekcja Escape.

Aby określić wartość tego atrybutu, należy użyć selektora CACMDQ z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *CommandLevel (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Poziom komend. Wskazuje poziom komend sterowania systemem obsługiwanych przez menedżer kolejek.

Poziom może przyjmować jedną z następujących wartości:

CML800
Poziom 800 komend sterowania systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i
  - 8.0

CML900
Poziom 900 komend sterowania systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i
  - 9.0

CML910
Poziom 910 komend sterowania systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i
  - 9.1

CML920
Poziom 920 komend sterowania systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i
  - 9.2

CML930
Poziom 930 komend sterowania systemem.

Ta wartość jest zwracana przez następujące aplikacje:

- IBM MQ for IBM i
  - Wersja 9.3

Zestaw komend sterujących systemu, który odpowiada konkretnej wartości atrybutu **CommandLevel** , różni się w zależności od wartości atrybutu **Platform** . Obie te komendy muszą zostać użyte do określenia, które z nich są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora IACMDL z wywołaniem MQINQ.

IBM i *ConfigurationEvent w systemie IBM i*

Określa, czy zdarzenia konfiguracji są generowane i wysyłane do systemu SYSTEM.ADMIN.CONFIG.EVENT .

Atrybut ConfigurationEvent może mieć jedną z następujących wartości:

- ERENA
- EVRDIS

Jeśli atrybut ConfigurationEvent ma wartość EVRENA, a niektóre komendy zostały pomyślnie wprowadzone przez komendę runmqsc lub PCF, zdarzenia konfiguracji są generowane i wysyłane do systemu SYSTEM.ADMIN.CONFIG.EVENT . Zdarzenia dla następujących komend są wysyłane, nawet jeśli komenda alter nie zmienia zaangażowanego obiektu. Komendy, dla których są generowane i wysyłane zdarzenia konfiguracyjne, to:

- ZDEFINIUJ/ZMIEŃ INFORMACJE O AUTORYZ.
- DEFINICJA/ZMIANA KANAŁU
- DEFINICJA/ZMIANA LISTY NAZW
- ZDEFINIUJ/ZMIEŃ PROCES
- DEFINE/ALTER QLOCAL (chyba że jest to tymczasowa kolejka dynamiczna)
- DEFINE/ALTER QMODEL/QALIAS/QREMOTE
- USUŃ INFORMACJE O AUTORYZ
- Usuń kanał
- USUŃ NAZWĘ
- Usuń proces
- DELETE QLOCAL (chyba że jest to tymczasowa kolejka dynamiczna)
- DELETE QMODEL/QALIAS/QREMOTE
- ALTER QMGR (chyba że atrybut CONFIGEV jest wyłączony i nie został zmieniony na włączony)
- ODŚWIEŻ MENEDŻERA KOLEJEK
- Wywołanie MQSET inne niż dla tymczasowej kolejki dynamicznej.

Zdarzenia nie są generowane (jeśli są włączone) w następujących okolicznościach:

- Wykonanie komendy lub wywołania MQSET nie powiodło się.
- Menedżer kolejek nie może umieścić komunikatu zdarzenia w kolejce zdarzeń. Komenda nadal powinna zakończyć się pomyślnie.
- Tymczasowe kolejki dynamiczne.
- Wewnętrzne zmiany atrybutów wykonywane bezpośrednio lub niejawnie (nie za pomocą komendy MQSET lub); ma to wpływ na TRIGGER, CURDEPTH, IPPROCS, OPPROCS, QDPHIEV, QDPLOEV, QDPMAXEV, QSVCIEV.
- Po zmianie kolejki zdarzeń konfiguracji, mimo że zostanie wygenerowany komunikat zdarzenia dla tej zmiany, gdy zostanie zażądane odświeżenie.
- Grupowanie zmian za pomocą komend REFRESH/RESET CLUSTER i RESUME/SUSPEND QMGR.
- Tworzenie lub usuwanie menedżera kolejek.

IBM i *DeadLetterQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa kolejki niedostarczonych komunikatów (niedostarczonych komunikatów).

Jest to nazwa kolejki zdefiniowana w menedżerze kolejek lokalnych. Komunikaty są wysyłane do tej kolejki, jeśli nie można ich skierować do właściwego miejsca docelowego.

Na przykład komunikaty są umieszczane w tej kolejce, gdy:

- Komunikat dociera do menedżera kolejek i jest przeznaczony dla kolejki, która nie została jeszcze zdefiniowana w tym menedżerze kolejek.
- Komunikat dociera do menedżera kolejek, ale kolejka, do której jest przeznaczony, nie może go odebrać, ponieważ prawdopodobnie:
  - Kolejka jest pełna
  - Żądania umieszczania są zablokowane
  - Węzeł wysyłający nie ma uprawnień do umieszczania komunikatów w kolejce

Aplikacje mogą również umieszczać komunikaty w kolejce niedostarczonych komunikatów.

Komunikaty raportu są traktowane w taki sam sposób, jak zwykłe komunikaty. Jeśli komunikat raportu nie może zostać dostarczony do kolejki docelowej (zwykle jest to kolejka określona w polu *MDRQ* w deskryptorze oryginalnego komunikatu), komunikat raportu jest umieszczany w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów).

Uwaga: Komunikaty, które przeszły okres ważności (patrz pole *MDEXP* opisane w sekcji "MQMD (deskryptor komunikatu) w systemie IBM i" na stronie 1141) **nie** są przesyłane do tej kolejki, gdy są usuwane. Jednak komunikat raportu o utracie ważności (ROEXP) jest nadal generowany i wysyłany do kolejki *MDRQ* na żądanie aplikacji wysyłającej.

Komunikaty nie są umieszczane w kolejce niedostarczonych komunikatów, jeśli aplikacja, która wysłała żądanie umieszczenia, została powiadomiona synchronicznie o problemie z kodem przyczyny zwróconym przez wywołanie MQPUT lub MQPUT1 (na przykład komunikat umieszczony w kolejce lokalnej, dla której żądania umieszczenia są zablokowane).

Dane komunikatów aplikacji w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów) mają czasami przedrostek w postaci struktury MQDLH. Ta struktura zawiera dodatkowe informacje wskazujące, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów). Więcej informacji na temat tej struktury zawiera sekcja "MQDLH (nagłówek niedostarczonego komunikatu) w systemie IBM i" na stronie 1094 .

Ta kolejka musi być kolejką lokalną z atrybutem **Usage** o wartości USNORM.

Jeśli kolejka niedostarczonych komunikatów (niedostarczonych komunikatów) nie jest obsługiwana przez menedżera kolejek lub nie została zdefiniowana, nazwa jest pusta. Wszystkie menedżery kolejek produktu IBM MQ obsługują kolejkę niedostarczonych komunikatów (niedostarczonych komunikatów), ale domyślnie nie jest ona zdefiniowana.

Jeśli kolejka niedostarczonych komunikatów (niedostarczonych komunikatów) nie jest zdefiniowana, jest pełna lub nie do użycia z innego powodu, komunikat, który zostałby do niej przesłany przez agenta kanału komunikatów, jest zachowywany w kolejce transmisji.

Aby określić wartość tego atrybutu, należy użyć selektora CADLQ z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

DefClusterXmitQueueTyp (10-cyfrowa liczba całkowita ze znakiem)

Atrybut `DefClusterXmitQueueType` określa, która kolejka transmisji jest domyślnie wybierana przez kanały nadawcze klastra, z których mają być wysyłane komunikaty do kanałów odbiorczych klastra.

Wartościami **DefClusterXmitQueueType** są `MQCLXQ_SCTQ` albo `MQCLXQ_CHANNEL`.

MQCLXQ_SCTQ

Wszystkie kanały nadawcze klastra wysyłają komunikaty z produktu `SYSTEM.CLUSTER.TRANSMIT.QUEUE`. Identyfikator `correlID` komunikatów umieszczonych w kolejce transmisji wskazuje, do którego kanału nadawczego klastra ma zostać przekazany komunikat.

Parametr `SNDJ` jest ustawiany podczas definiowania menedżera kolejek.

MQCLXQ_CHANNEL

Każdy kanał nadawczy klastra wysyła komunikaty z innej kolejki transmisji. Każda kolejka transmisji jest tworzona jako trwała kolejka dynamiczna z kolejki modelowej `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`.

Jeśli atrybut menedżera kolejek `DefClusterXmitQueueType`jest ustawiony na `CHANNEL`, Konfiguracja domyślna została zmieniona w taki sposób, że kanały nadawcze klastra zostały powiązane z poszczególnymi kolejkami transmisji klastra. Kolejki transmisji to trwałe kolejki dynamiczne utworzone na podstawie kolejki modelowej `SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE`. Każda kolejka transmisji jest powiązana z jednym kanałem nadawczym klastra. Ponieważ jeden kanał nadawczy klastra obsługuje kolejkę transmisji klastra, kolejka transmisji zawiera komunikaty dla tylko jednego menedżera kolejek w jednym klastrze. Istnieje możliwość skonfigurowania klastrów w taki sposób, aby każdy menedżer kolejek w klastrze zawierał tylko jedną kolejkę klastra. W takim przypadku ruch komunikatów z menedżera kolejek do każdej kolejki klastra jest przekazywany niezależnie z komunikatów do kolejki.

Aby wysłać zapytanie o wartość, należy wywołać funkcję `MQINQ`lub wysłać komendę Inquire Queue Manager (`MQCMD_INQUIRE_Q_MGR`) PCF, ustawiając selektor `MQIA_DEF_CLUSTER_XMIT_Q_TYPE` . Aby zmienić tę wartość, należy wysłać komendę PCF Change Queue Manager (`MQCMD_CHANGE_Q_MGR`), ustawiając selektor `MQIA_DEF_CLUSTER_XMIT_Q_TYPE` .

Odsyłacze pokrewne

Zmiana menedżera kolejek

Sprawdź menedżera kolejek

“MQINQ (zapytanie o atrybuty obiektu) w systemie IBM i” na stronie 1345

Wywołanie MQINQ zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierających atrybuty obiektu.

IBM i *DefXmitQName (48-bajtowy łańcuch znaków) w systemie IBM i*

Domyślna nazwa kolejki transmisji.

Jest to nazwa kolejki transmisji, która jest używana do transmisji komunikatów do zdalnych menedżerów kolejek, jeśli nie ma innego wskazania, której kolejki transmisji użyć.

Jeśli nie ma domyślnej kolejki transmisji, nazwa jest całkowicie pusta. Początkowa wartość tego atrybutu jest pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CADXQN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQN.

IBM i *DistLists (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Obsługa listy dystrybucyjnej.

Wskazuje, czy menedżer kolejek lokalnych obsługuje listy dystrybucyjne w wywołaniach MQPUT i MQPUT1 . Może mieć jedną z następujących wartości:

Obsługa DLSUPP
Obsługiwane listy dystrybucyjne.

DLNSUP
Listy dystrybucyjne nie są obsługiwane.

Aby określić wartość tego atrybutu, należy użyć selektora IADIST z wywołaniem MQINQ.

IBM i *InhibitEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia blokowania (blokowania pobierania i blokowania umieszczania).

Może mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie i wydajność.

Aby określić wartość tego atrybutu, należy użyć selektora IAINHE z wywołaniem MQINQ.

IBM i *LocalEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są lokalne zdarzenia błędów.

Jest to jedna z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń .

Aby określić wartość tego atrybutu, należy użyć selektora IALCLE z wywołaniem MQINQ.

IBM i *LoggerEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia programu rejestrującego odtwarzania.

Może mieć jedną z następujących wartości:

WŁĄCZONY
Generowane są zdarzenia programu rejestrującego.

WYŁĄCZONE
Zdarzenia programu rejestrującego nie są generowane. Jest to początkowa wartość domyślna menedżerów kolejek.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie i wydajność.

IBM i *MaxHandles (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalna liczba uchwytów.

Jest to maksymalna liczba otwartych uchwytów, które mogą być używane jednocześnie przez jedno zadanie. Każde pomyślne wywołanie MQOPEN dla pojedynczej kolejki (lub dla obiektu, który nie jest kolejką) używa jednego uchwytu. Ten uchwyt staje się dostępny do ponownego wykorzystania po zamknięciu obiektu. Jeśli jednak lista dystrybucyjna jest otwarta, każda kolejka na liście dystrybucyjnej ma przydzielony osobny uchwyt, dzięki czemu wywołanie MQOPEN używa tylu uchwytów, ile jest kolejek

na liście dystrybucyjnej. Należy to wziąć pod uwagę przy podejmowaniu decyzji o odpowiedniej wartości parametru *MaxHandles*.

Wywołanie MQPUT1 wykonuje wywołanie MQOPEN w ramach przetwarzania. W wyniku tego funkcja MQPUT1 używa tylu uchwytów, ile może użyć komenda MQOPEN, ale uchwyty są używane tylko przez czas trwania wywołania MQPUT1 .

Wartość należy do zakresu od 1 do 999 999 999. W systemie IBM iwartością domyślną jest 256.

Aby określić wartość tego atrybutu, należy użyć selektora IAMHND z wywołaniem MQINQ.

IBM i *MaxMsgDługość (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalna długość komunikatu w bajtach.

Jest to długość najdłuższego komunikatu *fizycznego* , który może być obsługiwany przez menedżer kolejek. Jednak ze względu na to, że atrybut menedżera kolejek **MaxMsgLength** można ustawić niezależnie od atrybutu kolejki **MaxMsgLength** , najdłuższy komunikat fizyczny, który można umieścić w kolejce, jest mniejszą z tych dwóch wartości.

Jeśli menedżer kolejek obsługuje segmentację, aplikacja może umieścić w programie MQMD komunikat *logiczny* , który jest dłuższy niż mniejsza z dwóch atrybutów **MaxMsgLength** , ale tylko wtedy, gdy w aplikacji określono flagę MFSEGA. Jeśli ta opcja jest określona, górny limit długości komunikatu logicznego wynosi 999 999 999 bajtów, ale zwykle ograniczenia zasobów narzucone przez system operacyjny lub środowisko, w którym działa aplikacja, spowodują obniżenie limitu.

Dolny limit dla atrybutu **MaxMsgLength** wynosi 32 kB (32 768 bajtów). W systemie IBM imaksymalna długość komunikatu wynosi 100 MB (104 857 600 bajtów).

Aby określić wartość tego atrybutu, należy użyć selektora IAMLEN z wywołaniem MQINQ.

IBM i *MaxPriority (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalny priorytet.

Jest to maksymalny priorytet komunikatu obsługiwany przez menedżer kolejek. Priorytety należą do zakresu od zera (najniższy) do *MaxPriority* (najwyższy).

Aby określić wartość tego atrybutu, należy użyć selektora IAMPRI z wywołaniem MQINQ.

IBM i *MaxUncommittedKomunikaty (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Maksymalna liczba niezatwierdzonych komunikatów w jednostce pracy.

Jest to maksymalna liczba niezatwierdzonych komunikatów, które mogą istnieć w jednostce pracy. Liczba niezatwierdzonych komunikatów jest sumą następujących wartości od momentu rozpoczęcia bieżącej jednostki pracy:

- Komunikaty umieszczane przez aplikację z opcją PMSYP
- Komunikaty pobrane przez aplikację z opcją GMSYP
- Komunikaty wyzwalacza i komunikaty raportu COA wygenerowane przez menedżer kolejek dla komunikatów umieszczonych za pomocą opcji PMSYP
- Komunikaty raportu COD wygenerowane przez menedżera kolejek dla komunikatów pobranych z opcją GMSYP

Następujące komunikaty nie są liczone jako niezatwierdzone:

- Komunikaty umieszczone lub pobrane przez aplikację poza jednostką pracy
- Komunikaty wyzwalacza lub COA/COD raportują komunikaty wygenerowane przez menedżera kolejek w wyniku umieszczenia lub pobrania komunikatów poza jednostką pracy
- Komunikaty raportu o utracie ważności wygenerowane przez menedżera kolejek (nawet jeśli wywołanie powodujące komunikat raportu o utracie ważności określały GMSYP)

- Komunikaty zdarzeń wygenerowane przez menedżera kolejek (nawet jeśli wywołanie powodujące komunikat zdarzenia określały PMSYP lub GMSYP)

Uwaga:

1. Komunikaty raportu o wyjątkach są generowane przez agent kanału komunikatów (MCA) lub przez aplikację i są traktowane w taki sam sposób, jak zwykłe komunikaty umieszczane lub pobierane przez aplikację.
2. Jeśli komunikat lub segment jest umieszczany z opcją PMSYP, liczba niezatwierdzonych komunikatów jest zwiększana o jeden, niezależnie od liczby komunikatów fizycznych rzeczywiście uzyskanych w wyniku umieszczenia. (Jeśli menedżer kolejek musi podzielić komunikat lub segment, może wystąpić więcej niż jeden komunikat fizyczny).
3. Gdy lista dystrybucyjna jest umieszczana z opcją PMSYP, liczba niezatwierdzonych komunikatów jest zwiększana o jeden *dla każdego wygenerowanego komunikatu fizycznego*. Liczba ta może być tak mała, jak jedna, lub tak duża, jak liczba miejsc docelowych na liście dystrybucyjnej.

Dolny limit dla tego atrybutu to 1; górny limit to 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora IAMUNC z wywołaniem MQINQ.

IBM i *PerformanceEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia związane z wydajnością.

PerformanceEvent może mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IAPFME z wywołaniem MQINQ.

IBM i *Platforma (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Platforma, na której jest uruchomiony menedżer kolejek.

Wskazuje system operacyjny, w którym działa menedżer kolejek. Wartość jest następująca:

PL400
IBM i.

IBM i *PubSubMode (10-cyfrowa liczba całkowita ze znakiem) on IBM i*

Określa, czy mechanizm publikowania/subskrypcji i umieszczony w kolejce interfejs publikowania/subskrypcji są uruchomione, co umożliwia aplikacjom publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania.

Może mieć jedną z następujących wartości:

PSMCP,
Mechanizm publikowania/subskrybowania działa. W związku z tym możliwe jest publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego. Umieszczony w kolejce interfejs publikowania/subskrypcji nie jest uruchomiony, dlatego nie są wykonywane żadne działania na komunikatach umieszczanych w kolejkach monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania. To ustawienie jest używane w celu zachowania zgodności z produktem WebSphere Message Broker V6 lub wcześniejszej używającym tego menedżera kolejek, ponieważ musi odczytywać te same kolejki, z których normalnie odczytywany jest umieszczony w kolejce interfejs publikowania/subskrypcji.

PSMDS
Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania nie działają. Dlatego nie jest możliwe publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego. Żadne komunikaty publikowania/subskrypcji, które są umieszczane w kolejkach monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania, nie są uwzględniane.

PSMEN
Mechanizm publikowania/subskrybowania oraz umieszczony w kolejce interfejs publikowania/subskrybowania działają. W związku z tym możliwe jest publikowanie/subskrybowanie przy użyciu aplikacyjnego interfejsu programistycznego i kolejek monitorowanych przez umieszczony w kolejce interfejs publikowania/subskrybowania. Jest to początkowa wartość domyślna menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora PSMODE z wywołaniem MQINQ.

IBM i *QMgrDesc (64-bitowy łańcuch znaków) w systemie IBM i*

Opis menedżera kolejek.

Jest to pole, którego można użyć jako komentarza opisowego. Zawartość pola nie ma znaczenia dla menedżera kolejek, ale menedżer kolejek może wymagać, aby pole zawierało tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole to może zawierać znaki DBCS (maksymalna długość pola wynosi 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), mogą one zostać niepoprawnie przetłumaczone, jeśli to pole zostanie wysłane do innego menedżera kolejek.

W systemie IBM iwartością domyślną jest wartość pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CAQMD z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQMD.

IBM i *QMgrIdentifier (48-bajtowy łańcuch znaków) w systemie IBM i*

Unikalny, wygenerowany wewnętrznie identyfikator menedżera kolejek.

Jest to wygenerowana wewnętrznie nazwa unikalna dla menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAQMID z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQMID.

IBM i *QMgrName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa menedżera kolejek.

Jest to nazwa lokalnego menedżera kolejek, czyli nazwa menedżera kolejek, z którym połączona jest aplikacja.

Pierwsze 12 znaków nazwy jest używanych do utworzenia unikalnego identyfikatora komunikatu (patrz pole *MDMID* opisane w sekcji “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141). Menedżery kolejek, które mogą komunikować się ze sobą, muszą mieć nazwy różniące się pierwszymi 12 znakami, aby identyfikatory komunikatów były unikalne w sieci menedżerów kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora CAQMN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQMN.

IBM i *RemoteEvent (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdalne zdarzenia błędów.

Jest to jedna z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IARMTE z wywołaniem MQINQ.

IBM i *RepositoryName (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa klastra, dla którego ten menedżer kolejek udostępnia usługi repozytorium.

Jest to nazwa klastra, dla którego ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę dla więcej niż jednego klastra, parametr *RepositoryNamelist* określa nazwę obiektu listy nazw, który identyfikuje klastry, a parametr *RepositoryName* jest pusty. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CARPN z wywołaniem MQINQ. Długość tego atrybutu jest określona przez LNQMN.

IBM i *RepositoryNamelist (48-bajtowy łańcuch znaków) w systemie IBM i*

Nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługi repozytorium.

Jest to nazwa obiektu listy nazw zawierającego nazwy klastrów, dla których ten menedżer kolejek udostępnia usługę menedżera repozytorium. Jeśli menedżer kolejek udostępnia tę usługę tylko dla jednego klastra, obiekt listy nazw zawiera tylko jedną nazwę. Alternatywnie można użyć parametru *RepositoryName* , aby określić nazwę klastra, w którym to przypadku wartość *RepositoryNamelist* jest pusta. Co najmniej jedna z wartości *RepositoryName* i *RepositoryNamelist* musi być pusta.

Aby określić wartość tego atrybutu, należy użyć selektora CARPNL z wywołaniem MQINQ. Długość tego atrybutu jest określona przez numer LNNLN.

IBM i *SSLEvent (łańcuch znaków) w systemie IBM i*

Określa, czy generowane są zdarzenia TLS.

Jest to jedna z następujących wartości:

- EVRENA (zdarzenie MQINQ/PCF/config) ENABLED (MQSC): generowane są zdarzenia TLS (to znaczy generowane jest zdarzenie RC2371).
- EVRDIS (zdarzenie MQINQ/PCF/config) DISABLED (MQSC): Zdarzenia TLS nie są generowane. Jest to początkowa wartość domyślna menedżera kolejek.

Aby określić wartość tego atrybutu, należy użyć selektora IASSLE z wywołaniem MQINQ.

IBM i *SSLKeyResetLiczba (liczba całkowita) w systemie IBM i*

Określa łączną liczbę niezaszyfrowanych bajtów, które są wysyłane i odbierane w ramach konwersacji TLS przed ponownym negocjowaniem klucza tajnego. Liczba bajtów obejmuje informacje sterujące wysłane przez agenta kanału komunikatów (MCA).

Ta wartość jest używana tylko przez adaptery MCA kanału TLS, które inicjują komunikację z tego menedżera kolejek (tj. agent MCA kanału nadawczego w parowaniu kanału nadawczego i odbiorczego).

Jeśli wartość tego atrybutu jest większa niż 0, a dla kanału są włączone pulsy kanału, klucz tajny jest również renegocjowany przed wysyłaniem lub odbieraniem danych zgodnie z pulsem kanału. Liczba bajtów do następnej ponownej negocjacji klucza tajnego, która zostanie zresetowana po każdej pomyślnej renegocjacji.

Wartość może być z zakresu od 0 do 999 999 999. Wartość 0 dla tego atrybutu wskazuje, że klucz tajny nigdy nie jest renegocjowany. Jeśli zostanie podana liczba operacji resetowania tajnego klucza TLS z zakresu od 1 bajtu do 32 kB, kanały TLS będą używać liczby operacji resetowania tajnego klucza o wielkości 32 kB. Ma to na celu uniknięcie kosztów przetwarzania nadmiernej liczby operacji resetowania klucza, które miałyby miejsce w przypadku małych wartości resetowania tajnego klucza TLS.

Jeśli serwer SSL jest menedżerem kolejek produktu IBM MQ i włączono zarówno resetowanie klucza tajnego, jak i puls kanału, renegocjacja następuje natychmiast po każdym pulsie kanału.

Aby określić wartość tego atrybutu, należy użyć selektora IASSRC z wywołaniem MQINQ.

IBM i *Zdarzenie StartStop(10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Określa, czy generowane są zdarzenia uruchomienia i zatrzymania.

Ten atrybut może mieć jedną z następujących wartości:

EVRDIS
Raportowanie zdarzeń jest wyłączone.

ERENA
Raportowanie zdarzeń jest włączone.

Więcej informacji na temat zdarzeń zawiera sekcja Monitorowanie zdarzeń.

Aby określić wartość tego atrybutu, należy użyć selektora IASSE z wywołaniem MQINQ.

IBM i *SyncPoint (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Dostępność punktu synchronizacji.

Wskazuje, czy lokalny menedżer kolejek obsługuje jednostki pracy i synchronizowanie z wywołaniami MQGET, MQPUT i MQPUT1 .

SPAVL
Dostępne jednostki pracy i synchronizowanie.

SPNAVL
Jednostki pracy i synchronizowanie nie są dostępne.

Aby określić wartość tego atrybutu, należy użyć selektora IASYNC z wywołaniem MQINQ.

IBM i *TraceRouteRejestrowanie (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Ta opcja określa, czy informacje o komunikatach są rejestrowane podczas ich przepływu przez menedżer kolejek.

Jest to jedna z następujących wartości:

- RECDD: nie jest dozwolone dopisywanie do komunikatów trasy śledzenia
- RECDQ: komunikaty są umieszczane w stałej kolejce nazwanej
- RECDM: określ przy użyciu komunikatu (jest to początkowe ustawienie domyślne)

Aby zapobiec pozostawaniu komunikatu trasy śledzenia w systemie, należy ustawić dla niego wartość utraty ważności większą od zera i określić opcję raportu RODISC. Aby zapobiec pozostawaniu komunikatów raportu lub odpowiedzi w systemie, należy ustawić opcję raportu ROPDAE. Więcej informacji na ten temat zawiera sekcja “Opcje raportu i flagi komunikatów w systemie IBM i” na stronie 1478.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGI z wywołaniem MQINQ.

IBM i *TreeLifeTime (10-cyfrowa liczba całkowita ze znakiem) on IBM i*

Czas życia (w sekundach) tematów nieadministracyjnych.

Tematy nieadministracyjne to te, które są tworzone, gdy aplikacja publikuje lub subskrybuje jako łańcuch tematu, który nie istnieje jako węzeł administracyjny. Kiedy w danym węźle nieadministracyjnym nie ma już żadnych aktywnych subskrypcji, ten parametr określa czas, przez jaki menedżer kolejek będzie oczekiwać przed usunięciem tego węzła. Tylko te tematy nieadministrowane, które są używane w ramach trwałej subskrypcji, przetrwają przetwarzanie wtórne menedżera kolejek.

Należy podać wartość z zakresu od 0 do 604 000. Wartość 0 oznacza, że tematy nieadministrowane nie są usuwane przez menedżer kolejek. Domyślna wartość początkowa menedżera kolejek to 1800.

Aby określić wartość tego atrybutu, należy użyć selektora IATRLFT z wywołaniem MQINQ.

IBM i *TriggerInterval (10-cyfrowa liczba całkowita ze znakiem) w systemie IBM i*

Odstęp czasu między komunikatami wyzwalacza.

Jest to przedział czasu (w milisekundach) używany do ograniczenia liczby komunikatów wyzwalacza. Ma to zastosowanie tylko wtedy, gdy parametr *TriggerType* ma wartość TTFRST. W takim przypadku komunikaty wyzwalacza są zwykle generowane tylko wtedy, gdy w kolejce pojawi się odpowiedni komunikat, a kolejka była wcześniej pusta. Jednak w pewnych okolicznościach można wygenerować dodatkowy komunikat wyzwalacza z wyzwalaniem TTFRST, nawet jeśli kolejka nie była pusta. Te dodatkowe komunikaty wyzwalacza nie są generowane częściej niż co *TriggerInterval* milisekund.

Więcej informacji na temat wyzwalania zawiera sekcja Wyzwalanie kanałów.

Wartość należy do zakresu od 0 do 999 999 999. Wartością domyślną jest 999 999 999.

Aby określić wartość tego atrybutu, należy użyć selektora IATRGI z wywołaniem MQINQ.

Aplikacje

W tej sekcji opisano przykładowe programy dostarczane z produktem IBM MQ for IBM i for RPG. Należy również zapoznać się z informacjami na temat budowania aplikacji wykonywalnych na podstawie napisanych programów.

Budowanie aplikacji

Publikacje IBM i zawierają opis sposobu budowania aplikacji wykonywalnych na podstawie programów napisanych przez użytkownika. W tym temacie opisano dodatkowe zadania oraz zmiany w zadaniach standardowych, które należy wykonać podczas budowania aplikacji IBM MQ for IBM i uruchamianych w systemie IBM i.

Oprócz kodowania wywołań MQI w kodzie źródłowym należy dodać odpowiednie instrukcje języka, aby dołączyć pliki kopii IBM MQ for IBM i dla języka RPG. Należy zapoznać się z zawartością tych plików; ich nazwy i krótki opis znajdują się w poniższym tekście.

IBM i *IBM MQ kopiowanie plików w systemie IBM i*

Program IBM MQ for IBM i udostępnia pliki kopii ułatwiające pisanie aplikacji w języku programowania RPG. Są one odpowiednie do użycia z zestawem narzędzi programistycznych WebSphere (5722 WDS) ILE RPG 4 Compiler.

Pliki kopii udostępniane przez program IBM MQ for IBM i w celu ułatwienia zapisu wyjść kanałów zostały opisane w sekcji Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów.

Nazwy plików kopii IBM MQ for IBM i dla języka RPG mają przedrostek CMQ. Mają one przyrostek G lub H. Istnieją oddzielne pliki kopii zawierające stałe nazwane i jeden plik dla każdej struktury. Lista plików kopii znajduje się w sekcji “Uwagi dotyczące języka” na stronie 1037.

Uwaga: Dla ILE RPG/400, są one dostarczane jako podzbiory zbioruQRPGLESRC w bibliotece QMQM.

Deklaracje struktur nie zawierają instrukcji DS . Dzięki temu aplikacja może zadeklarować strukturę danych (lub strukturę danych o wielu wystąpieniach), kodując instrukcję DS i używając instrukcji /COPY do skopiowania pozostałej części deklaracji:

Dla ILE RPG/400 instrukcją jest:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7
D* Declare an MQMD data structure
D MQMD       DS
D/COPY CMQMDG
```

Przygotowywanie programów do uruchomienia

Aby utworzyć wykonywalną aplikację IBM MQ for IBM i , należy skompilować napisany kod źródłowy.

W tym celu dla ILE RPG/400można użyć typowych komend IBM i , CRTRPGMOD i CRTPGM.

Po utworzeniu wartości *MODULE należy określić parametr `BNDSRVPGM(QMQM/LIBMQM)` w komendzie CRTPGM. Obejmuje to różne procedury IBM MQ w programie.

Upewnij się, że biblioteka zawierająca zbiory kopii (QMQM) znajduje się na liście bibliotek podczas wykonywania kompilacji.

Więcej informacji na temat zagadnień związanych z programowaniem, w tym na temat trybów klienta, zawiera sekcja “Uwagi dotyczące języka” na stronie 1037.

Interfejsy do zewnętrznego menedżera punktów synchronizacji IBM i

Produkt IBM MQ for IBM i używa rodzimej kontroli transakcji IBM i jako zewnętrznego koordynatora punktu synchronizacji.

Więcej informacji na temat możliwości kontroli transakcji systemu IBM izawiera publikacja *IBM i Programming: Backup and Recovery Guide* .

Aby uruchomić narzędzia kontroli transakcji systemu IBM i , należy użyć komendy systemowej STRCMTCTL. Aby zakończyć kontrolę transakcji, należy użyć komendy systemowej ENDCMTCTL.

Uwaga: Wartością domyślną parametru *Zasięg definicji kontroli transakcji* jest *ACTGRP. Musi być ona zdefiniowana jako *JOB dla systemu IBM MQ for IBM i. Na przykład:

```
STRCMTCTL LCKLVL(*ALL) CMTSCOPE(*JOB)
```

W przypadku wywołania MQPUT, MQPUT1lub MQGET z określeniem wartości PMSYP lub GMSYP, po uruchomieniu kontroli transakcji program IBM MQ for IBM i dodaje się do definicji kontroli transakcji jako zasób kontroli transakcji API. Jest to zazwyczaj pierwsze takie wywołanie w zadaniu. Chociaż istnieją jakiekolwiek zasoby kontroli transakcji API zarejestrowane w ramach określonej definicji kontroli transakcji, nie można zakończyć kontroli transakcji dla tej definicji.

Produkt IBM MQ for IBM i usuwa swoją rejestrację jako zasób zatwierdzania transakcji API po rozłączeniu się z menedżerem kolejek, pod warunkiem, że w bieżącej jednostce pracy nie ma oczekujących operacji MQI.

W przypadku rozłączenia z menedżerem kolejek, gdy w bieżącej jednostce pracy istnieją oczekujące operacje MQPUT, MQPUT1lub MQGET, produkt IBM MQ for IBM i pozostaje zarejestrowany jako zasób zatwierdzania API, dzięki czemu jest powiadamiany o następnym zatwierdzeniu lub wycofaniu zmian. Po osiągnięciu następnego punktu synchronizacji program IBM MQ zatwierdza lub wycofuje zmiany zgodnie z wymaganiami. Aplikacja może rozłączyć się i ponownie nawiązać połączenie z menedżerem kolejek podczas aktywnej jednostki pracy oraz wykonać dalsze operacje MQGET i MQPUT w obrębie tej samej jednostki pracy (jest to rozłączenie w toku).

Jeśli dla tej definicji kontroli transakcji zostanie wydana komenda systemowa ENDCMTCTL, zostanie wyświetlony komunikat CPF8355 , wskazujący, że oczekujące zmiany były aktywne. Komunikat ten pojawia się również w protokole zadania po zakończeniu zadania. Aby tego uniknąć, należy zatwierdzić lub wycofać wszystkie oczekujące operacje produktu IBM MQ oraz rozłączyć się z menedżerem kolejek. Dlatego użycie komend COMMIT lub ROLLBACK przed ENDCMTCTL powinno umożliwić pomyślne zakończenie kontroli transakcji.

Jeśli kontrola transakcji IBM i jest używana jako zewnętrzny koordynator punktu synchronizacji, wywołania MQCMIT, MQBACK i MQBEGIN mogą nie być wykonywane. Wywołania tych funkcji kończą się niepowodzeniem z kodem przyczyny RC2012.

Aby zatwierdzić lub wycofać zmiany (czyli wycofać zmiany) jednostki pracy, należy użyć jednego z języków programowania, który obsługuje kontrolę transakcji. Na przykład:

- Komendy CL: COMMIT i ROLLBACK

- Funkcje programistyczne ILE C: _Rcommit i _Rrollback
- RPG/400: COMMIT i ROLBK
- COBOL/400: zatwierdzanie i wycofywanie zmian

Punkty synchronizacji w produkcie CICS dla aplikacji IBM i

IBM MQ for IBM i uczestniczy w jednostkach pracy z CICS. Interfejsu MQI w aplikacji CICS można używać w celu umieszczania i pobierania komunikatów w bieżącej jednostce pracy.

Za pomocą komendy EXEC CICS SYNCPOINT można ustanowić punkt synchronizacji, który obejmuje operacje IBM MQ for IBM i . Aby wycofać wszystkie zmiany do poprzedniego punktu synchronizacji, można użyć komendy EXEC CICS SYNCPOINT ROLLBACK.

Jeśli w aplikacji CICS używane są opcje MQPUT, MQPUT1lub MQGET z opcją PMSYP lub GMSYP, nie można wylogować się CICS , dopóki program IBM MQ for IBM i nie usunie swojej rejestracji jako zasobu zatwierdzania transakcji API. Dlatego przed rozłączeniem się z menedżerem kolejek należy zatwierdzić lub wycofać wszystkie oczekujące operacje umieszczania lub pobierania. Umożliwi to wylogowanie się z programu CICS.

Programy przykładowe w systemie IBM i

W tym temacie opisano przykładowe programy dostarczane z produktem IBM MQ for IBM i for RPG. Przykłady przedstawiają typowe zastosowania interfejsu kolejki komunikatów (Message Queue Interface-MQI).

Przykłady nie mają na celu zademonstrowania ogólnych technik programowania, dlatego pominięto pewne sprawdzanie błędów, które można włączyć do programu produkcyjnego. Przykłady te są jednak odpowiednie do użycia jako podstawa dla własnych programów kolejkowania komunikatów.

Kod źródłowy wszystkich przykładów jest dostarczany wraz z produktem. Źródło to zawiera komentarze, które wyjaśniają techniki kolejkowania komunikatów przedstawione w programach.

Istnieje jeden zestaw przykładowych programów ILE:

1. **Programy używające protokołu wywołań MQI (static bound calls)**

   Źródło istnieje w QMQMSAMP/QRPGLESRC. Elementy mają nazwę AMQ3xxx4, gdzie xxx oznacza przykładową funkcję. W QMQM/QRPGLESRC istnieją podzbiory kopii. Każda nazwa elementu ma przyrostek G lub H.

Tabela 811 na stronie 1460 zawiera pełną listę przykładowych programów dostarczonych z produktem IBM MQ for IBM ioraz nazwy programów w każdym z obsługiwanych języków programowania. Należy zauważyć, że wszystkie ich nazwy zaczynają się od przedrostka AMQ, czwarty znak w nazwie wskazuje język programowania.

Tabela 811. Nazwy programów przykładowych

	RPG (ILE)
Umieść próbki	AMQ3PUT4
Przeglądaj przykłady	AMQ3GBR4
Pobierz przykłady	AMQ3GET4
Przykłady żądań	AMQ3REQ4
Przykłady echa	AMQ3ECH4
Sprawdź przykłady	AMQ3INQ4
Ustaw próbki	AMQ3SET4
Przykład monitora wyzwalacza	AMQ3TRG4

Tabela 811. Nazwy programów przykładowych (kontynuacja)	
	RPG (ILE)
Przykład serwera wyzwalaczy	AMQ3SRV4

Oprócz tych opcji, przykładowa opcja IBM MQ for IBM i zawiera przykładowy plik danych AMQSDATA, który może być używany jako dane wejściowe dla niektórych programów przykładowych i przykładowych programów CL demonstrujących zadania administracyjne. Przykłady CL zostały opisane w sekcji Administrowanie IBM i . Za pomocą przykładowego programu CL można utworzyć kolejki do użycia z przykładowymi programami opisanymi w tym temacie.

Informacje na temat uruchamiania programów przykładowych zawiera sekcja "Przygotowywanie i uruchamianie programów przykładowych w systemie IBM i" na stronie 1462.

Funkcje demonstrowane w przykładowych programach w systemie IBM i

Tabela przedstawiająca techniki demonstrowane przez programy przykładowe IBM MQ for IBM i .

Niektóre techniki występują w więcej niż jednym programie przykładowym, ale tylko jeden program jest wymieniony w tabeli. Wszystkie przykłady otwierają i zamykają kolejki przy użyciu wywołań MQOPEN i MQCLOSE, dlatego te techniki nie są wymienione oddzielnie w tabeli.

Tabela 812. Przykładowe programy demonstrujące użycie interfejsu MQI	
Technika	**RPG (ILE)**
Korzystanie z wywołań MQCONN i MQDISC	AMQ3ECH4 lub AMQ3INQ4
Niejawne łączenie i rozłączanie	AMQ3PUT4
Umieszczanie komunikatów przy użyciu wywołania MQPUT	AMQ3PUT4
Umieszczanie pojedynczego komunikatu przy użyciu wywołania MQPUT1	AMQ3ECH4 lub AMQ3INQ4
Odpowiadanie na komunikat żądania	AMQ3INQ4
Pobieranie komunikatów (bez oczekiwania)	AMQ3GBR4
Pobieranie komunikatów (oczekiwanie z limitem czasu)	AMQ3GET4
Pobieranie komunikatów (z konwersją danych)	AMQ3ECH4
Przeglądanie kolejki	AMQ3GBR4
Korzystanie ze współużytkowanej kolejki wejściowej	AMQ3INQ4
Korzystanie z wyłącznej kolejki wejściowej	AMQ3REQ4
Korzystanie z wywołania MQINQ	AMQ3INQ4
Korzystanie z wywołania MQSET	AMQ3SET4
Korzystanie z kolejki odpowiedzi	AMQ3REQ4
Wysyłanie żądań komunikatów o wyjątkach	AMQ3REQ4
Akceptowanie obciętego komunikatu	AMQ3GBR4
Używanie rozstrzygniętej nazwy kolejki	AMQ3GBR4
Przetwarzanie wyzwalacza	AMQ3SRV4 lub AMQ3TRG4

Uwaga: Wszystkie programy przykładowe generują zbiór buforowy, który zawiera wyniki przetwarzania.

Przygotowywanie i uruchamianie programów przykładowych w systemie IBM i

Przed uruchomieniem przykładowych programów IBM MQ for IBM i należy je skompilować w taki sam sposób, jak inne aplikacje IBM MQ for IBM i . W tym celu można użyć komend IBM i CRTRPGMOD i CRTPGM.

Podczas tworzenia programów AMQ3xxx4 należy określić BNDSRVPGM (QMQM/LIBMQM) w komendzie CRTPGM. Obejmuje to różne procedury IBM MQ w programie.

Programy przykładowe są udostępniane w bibliotece QMQMSAMP jako podzbiory QRPGLESRC. Używają one zbiorów kopii udostępnionych w bibliotece QMQM, dlatego należy upewnić się, że ta biblioteka znajduje się na liście bibliotek podczas kompilowania. Kompilator języka RPG generuje komunikaty informacyjne, ponieważ przykłady nie używają wielu zmiennych zadeklarowanych w plikach kopii.

Uruchamianie programów przykładowych

Podczas uruchamiania przykładów można używać własnych kolejek lub można skompilować i uruchomić komendę AMQSAMP4 w celu utworzenia kolejek przykładowych. Źródło tego programu jest dostarczane w zbiorze QCLSRC w bibliotece QMQMSAMP. Można ją skompilować przy użyciu komendy CRTCLPGM.

Aby wywołać jeden z przykładowych programów, należy użyć następującej komendy:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Gdzie `Queue_Name` i `Queue_Manager_Name` muszą mieć długość 48 znaków, co jest osiągane przez dopełnienie `Queue_Name` i `Queue_Manager_Name` wymaganą liczbą odstępów.

W przypadku przykładowych programów Inquire i Set przykładowe definicje utworzone przez program AMQSAMP4 powodują wyzwolenie wersji C tych przykładów. Aby wyzwolić wersje języka RPG, należy zmienić definicje procesów SYSTEM.SAMPLE.ECHOPROCESS i SYSTEM.SAMPLE.INQPROCESS i SYSTEM.SAMPLE.SETPROCESS. Można użyć komendy CHGMQMPRC (opisanej w sekcji Zmiana procesu MQ (Change MQ Process-CHGMQMPRC)). w tym celu należy dokonać edycji lub uruchomić komendę AMQSAMP4 z alternatywną definicją.

Przykładowy program Put w systemie IBM i

Program przykładowy Put, AMQ3PUT4, umieszcza komunikaty w kolejce przy użyciu wywołania MQPUT.

Aby uruchomić program, wywołaj program i podaj nazwę kolejki docelowej jako parametr programu. Program umieszcza zestaw stałych komunikatów w kolejce; komunikaty te są pobierane z bloku danych na końcu kodu źródłowego programu. Przykładowy program put to AMQ3PUT4 w bibliotece QMQMSAMP.

Za pomocą tego programu przykładowego komenda jest następująca:

```
CALL PGM(QMQMSAMP/AMQ3PUT4) PARM('Queue_Name','Queue_Manager_Name')
```

Gdzie `Queue_Name` i `Queue_Manager_Name` muszą mieć długość 48 znaków, co jest osiągane przez dopełnienie `Queue_Name` i `Queue_Manager_Name` wymaganą liczbą odstępów.

Projekt programu przykładowego Put

Program używa wywołania MQOPEN z opcją OOOUT do otwarcia kolejki docelowej na potrzeby umieszczania komunikatów. Wyniki są wyprowadzane do zbioru buforowego. Jeśli nie można otworzyć kolejki, program zapisuje komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN. Aby zachować prostotę programu, w tym przypadku i w kolejnych wywołaniach MQI program używa wartości domyślnych dla wielu opcji.

Dla każdego wiersza danych zawartego w kodzie źródłowym program odczytuje tekst do buforu i używa wywołania MQPUT do utworzenia komunikatu datagramu zawierającego tekst w tym wierszu. Działanie programu jest kontynuowane do momentu, gdy zostanie osiągnięty koniec wejścia lub wywołanie MQPUT nie powiedzie się. Jeśli program osiągnie koniec danych wejściowych, zamyka kolejkę za pomocą wywołania MQCLOSE.

Przykładowy program Przeglądaj w systemie IBM i

Przykładowy program przeglądania (AMQ3GBR4) przegląda komunikaty w kolejce przy użyciu wywołania MQGET.

Program wczytuje kopie wszystkich komunikatów w kolejce określonej podczas wywoływania programu; komunikaty pozostają w kolejce. Można użyć dostarczonej kolejki SYSTEM.SAMPLE.LOCAL-uruchom przykładowy program Put jako pierwszy, aby umieścić niektóre komunikaty w kolejce. Można użyć kolejki SYSTEM.SAMPLE.ALIAS, która jest aliasem tej samej kolejki lokalnej. Działanie programu jest kontynuowane do momentu osiągnięcia końca kolejki lub niepowodzenia wywołania MQI.

Przykład komendy wywołującej program RPG:

```
CALL PGM(QMQMSAMP/AMQ3GBR4) PARM('Queue_Name','Queue_Manager_Name')
```

Gdzie `Queue_Name` i `Queue_Manager_Name` muszą mieć długość 48 znaków, co jest osiągane przez dopełnienie `Queue_Name` i `Queue_Manager_Name` wymaganą liczbą odstępów. Oznacza to, że w przypadku korzystania z pliku SYSTEM.SAMPLE.LOCAL jako kolejka docelowa musi zawierać 29 znaków odstępu.

Projekt przykładowego programu przeglądania

Program otwiera kolejkę docelową za pomocą wywołania MQOPEN z opcją OOBRW. Jeśli nie można otworzyć kolejki, program zapisuje komunikat o błędzie w swoim zbiorze buforowym, który zawiera kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego komunikatu w kolejce program używa wywołania MQGET do skopiowania komunikatu z kolejki, a następnie wyświetla dane zawarte w komunikacie. Wywołanie MQGET używa następujących opcji:

GMBRWN
: Po wywołaniu MQOPEN kursor przeglądania jest umieszczany logicznie przed pierwszym komunikatem w kolejce, dlatego ta opcja powoduje zwrócenie *pierwszego* komunikatu przy pierwszym wywołaniu.

GMNWT,
: Program nie czeka, jeśli w kolejce nie ma żadnych komunikatów.

GMATM,
: Wywołanie MQGET określa bufor o stałej wielkości. Jeśli komunikat jest dłuższy niż ten bufor, program wyświetla obcięty komunikat wraz z ostrzeżeniem, że komunikat został obcięty.

Program demonstruje, w jaki sposób należy wyczyścić pola *MDMID* i *MDCID* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie to ustawia te pola na wartości zawarte w pobieranym komunikacie. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej są przechowywane w kolejce.

Program kontynuuje działanie do końca kolejki; w tym przypadku wywołanie MQGET zwraca kod przyczyny RC2033 (brak dostępnego komunikatu), a program wyświetla komunikat ostrzegawczy. Jeśli wywołanie MQGET nie powiedzie się, program zapisze komunikat o błędzie zawierający kod przyczyny w zbiorze buforowym.

Następnie program zamyka kolejkę za pomocą wywołania MQCLOSE.

Przykładowy program Get w systemie IBM i

Program przykładowy Get, AMQ3GET4, pobiera komunikaty z kolejki za pomocą wywołania MQGET.

Po wywołaniu programu usuwa on komunikaty z określonej kolejki. Można użyć dostarczonej kolejki SYSTEM.SAMPLE.LOCAL-uruchom przykładowy program Put jako pierwszy, aby umieścić niektóre komunikaty w kolejce. Można użyć pliku SYSTEM.SAMPLE.ALIAS , która jest aliasem tej samej kolejki lokalnej. Działanie programu jest kontynuowane do momentu, gdy kolejka będzie pusta lub wywołanie MQI nie powiedzie się.

Przykład komendy wywołującej program RPG:

```
CALL PGM(QMQMSAMP/AMQ3GET4) PARM('Queue_Name','Queue_Manager_Name')
```

gdzie Queue_Name i Queue_Manager_Name muszą mieć długość 48 znaków, co jest osiągane przez dopełnienie znaków Queue_Name i Queue_Manager_Name wymaganą liczbą odstępów. Oznacza to, że w przypadku korzystania z pliku SYSTEM.SAMPLE.LOCAL jako kolejka docelowa musi zawierać 29 znaków odstępu.

Projekt przykładowego programu Get

Program otwiera kolejkę docelową na potrzeby pobierania komunikatów; używa wywołania MQOPEN z opcją OOINPQ. Jeśli nie można otworzyć kolejki, program zapisuje komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN w swoim zbiorze buforowym.

Dla każdego komunikatu w kolejce program używa wywołania MQGET do usunięcia komunikatu z kolejki, a następnie wyświetla dane zawarte w komunikacie. Wywołanie MQGET używa opcji GMWT z interwałem oczekiwania (*GMWI*) wynoszącym 15 sekund, tak aby program oczekiwał przez ten okres, jeśli w kolejce nie ma komunikatu. Jeśli przed upływem tego okresu nie nadejdzie żaden komunikat, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2033 (brak dostępnego komunikatu).

Program demonstruje, w jaki sposób należy wyczyścić pola *MDMID* i *MDCID* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie to ustawia te pola na wartości zawarte w pobieranym komunikacie. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej są przechowywane w kolejce.

Wywołanie MQGET określa bufor o stałej wielkości. Jeśli komunikat jest dłuższy niż ten bufor, wywołanie nie powiedzie się i program zostanie zatrzymany.

Program będzie kontynuował działanie, dopóki wywołanie MQGET nie zwróci kodu przyczyny RC2033 (brak dostępnego komunikatu) lub wywołanie MQGET nie powiedzie się. Jeśli wywołanie nie powiedzie się, program wyświetli komunikat o błędzie zawierający kod przyczyny.

Następnie program zamyka kolejkę za pomocą wywołania MQCLOSE.

Przykładowy program żądania w systemie IBM i

Przykładowy program żądania AMQ3REQ4demonstruje przetwarzanie klient/serwer. Przykładem jest klient, który umieszcza komunikaty żądań w kolejce przetwarzanej przez program serwera. Oczekuje on na umieszczenie komunikatu odpowiedzi w kolejce odpowiedzi przez program serwera.

Przykład żądania umieszcza serię komunikatów żądań w kolejce przy użyciu wywołania MQPUT. Te komunikaty określają plik SYSTEM.SAMPLE.REPLY as the reply-to queue (Odpowiedź jako kolejka odpowiedzi). Program oczekuje na komunikaty odpowiedzi, a następnie je wyświetla. Odpowiedzi są wysyłane tylko wtedy, gdy kolejka docelowa (która będzie zwana *kolejką serwera*) jest przetwarzana przez aplikację serwera lub jeśli aplikacja jest wyzwalana w tym celu (programy przykładowe Inquire i Set są przeznaczone do wyzwolenia). Próbka oczekuje 5 minut na nadejście pierwszej odpowiedzi (aby umożliwić wyzwolenie aplikacji serwera) i 15 sekund na kolejne odpowiedzi, ale może się zakończyć bez uzyskania żadnych odpowiedzi.

Aby uruchomić program, wywołaj program i podaj nazwę kolejki docelowej jako parametr programu. Program umieszcza zestaw stałych komunikatów w kolejce; komunikaty te są pobierane z bloku danych na końcu kodu źródłowego programu.

Projekt przykładowego programu żądania

Program otwiera kolejkę serwera, dzięki czemu może umieszczać komunikaty. Używa wywołania MQOPEN z opcją OOOUT. Jeśli nie można otworzyć kolejki, program wyświetla komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Następnie program otwiera kolejkę odpowiedzi o nazwie SYSTEM.SAMPLE.REPLY , aby mogła otrzymać komunikaty odpowiedzi. W tym celu program używa wywołania MQOPEN z opcją OOINPX. Jeśli nie można

otworzyć kolejki, program wyświetla komunikat o błędzie zawierający kod przyczyny zwrócony przez wywołanie MQOPEN.

Dla każdego wiersza danych wejściowych program odczytuje tekst do buforu i używa wywołania MQPUT do utworzenia komunikatu żądania zawierającego tekst tego wiersza. W tym wywołaniu program używa opcji raportu ROEXCD do żądania, aby wszystkie wysłane komunikaty raportu dotyczące komunikatu żądania zawierały pierwsze 100 bajtów danych komunikatu. Działanie programu jest kontynuowane do momentu, gdy zostanie osiągnięty koniec wejścia lub wywołanie MQPUT nie powiedzie się.

Następnie program używa wywołania MQGET do usunięcia komunikatów odpowiedzi z kolejki i wyświetla dane zawarte w odpowiedziach. Wywołanie MQGET używa opcji GMWT, określając interwał oczekiwania (*GMWI*) wynoszący 5 minut dla pierwszej odpowiedzi (w celu umożliwienia wyzwolenia aplikacji serwera) i 15 sekund dla kolejnych odpowiedzi. Program oczekuje przez te okresy, jeśli w kolejce nie ma żadnego komunikatu. Jeśli przed upływem tego okresu nie nadejdzie żaden komunikat, wywołanie nie powiedzie się i zostanie zwrócony kod przyczyny RC2033 (brak dostępnego komunikatu). Wywołanie używa również opcji GMATM, więc komunikaty dłuższe niż zadeklarowana wielkość buforu są obcinane.

Program demonstruje, w jaki sposób należy wyczyścić pola *MDMID* i *MDCOD* struktury MQMD po każdym wywołaniu MQGET, ponieważ wywołanie to ustawia te pola na wartości zawarte w pobieranym komunikacie. Usunięcie zaznaczenia tych pól oznacza, że kolejne wywołania MQGET pobierają komunikaty w kolejności, w jakiej są przechowywane w kolejce.

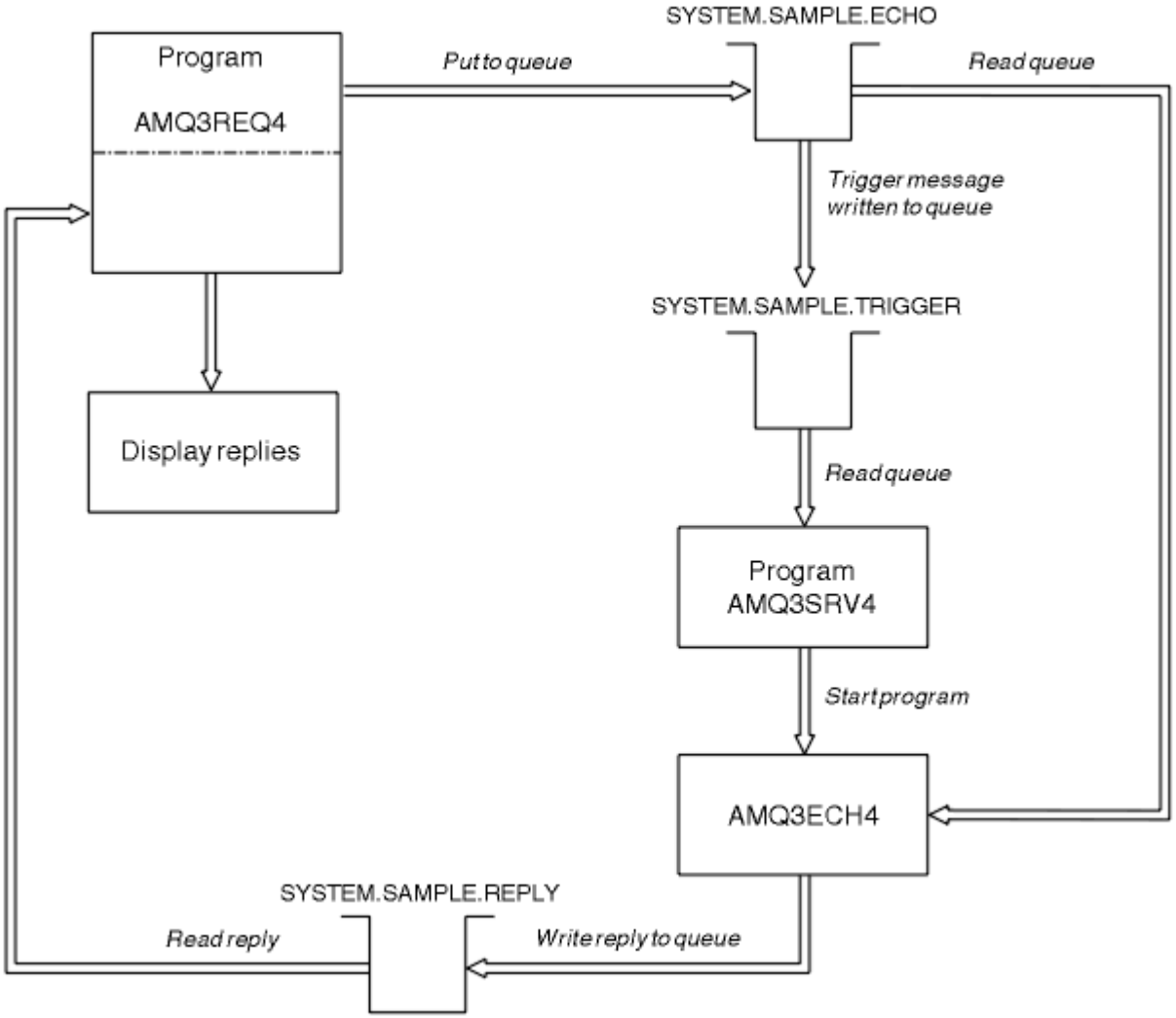
Program będzie kontynuował działanie, dopóki wywołanie MQGET nie zwróci kodu przyczyny RC2033 (brak dostępnego komunikatu) lub wywołanie MQGET nie powiedzie się. Jeśli wywołanie nie powiedzie się, program wyświetli komunikat o błędzie zawierający kod przyczyny.

Następnie program zamyka zarówno kolejkę serwera, jak i kolejkę odpowiedzi za pomocą wywołania MQCLOSE. Tabela 813 na stronie 1465 przedstawia zmiany w programie przykładowym Echo, które są niezbędne do uruchomienia programów przykładowych Inquire i Set.

Uwaga: Szczegółowe informacje na temat przykładowego programu Echo znajdują się w skorowidzu.

Tabela 813. Szczegóły programu przykładowego klient/serwer

Nazwa programu	**Kolejka SYSTEM/SAMPLE**	**Program został uruchomiony**
Echo	ECHO	AMQ3ECH4
Zapytania	INQ	AMQ3INQ4
Ustaw	SET	AMQ3SET4



Rysunek 9. Przykładowy schemat blokowy programu klient/serwer (Echo)

IBM i *Używanie wyzwalania z próbką żądania w systemie IBM i*

Aby uruchomić przykład przy użyciu wyzwalania, uruchom program serwera wyzwalaczy, AMQ3SRV4, dla wymaganej kolejki inicjowania w jednym zadaniu, a następnie uruchom komendę AMQ3REQ4 w innym zadaniu.

Oznacza to, że serwer wyzwalacza jest gotowy, gdy przykładowy program żądania wysyła komunikat.

Uwaga:

1. Przykłady używają systemowej kolejki SAMPLE TRIGGER jako kolejki inicjującej dla pliku SYSTEM.SAMPLE.ECHO, SYSTEM.SAMPLE.INQlub SYSTEM.SAMPLE.SET kolejek lokalnych. Alternatywnie można zdefiniować własną kolejkę inicjującą.
2. Przykładowe definicje utworzone przez program AMQSAMP4 powodują wyzwolenie wersji C przykładu. Aby wyzwolić wersję języka RPG, należy zmienić definicje procesów SYSTEM.SAMPLE.ECHOPROCESS i SYSTEM.SAMPLE.INQPROCESS i SYSTEM.SAMPLE.SETPROCESS. W tym celu można użyć komendy CHGMQMPRC (więcej szczegółów zawiera sekcja Zmiana procesu MQ (CHGMQMPRC)) lub zmodyfikować i uruchomić własną wersję programu AMQSAMP4.
3. Należy skompilować program serwera wyzwalaczy ze źródła podanego w QMQMSAMP/QRPGLESRC.

W zależności od uruchamianego procesu wyzwalacza należy wywołać komendę AMQ3REQ4 z parametrem określającym komunikaty żądań, które mają zostać umieszczone w jednej z następujących przykładowych kolejek serwera:

- SYSTEM.SAMPLE.ECHO (dla przykładowych programów Echo)
- SYSTEM.SAMPLE.INQ (dla programów przykładowych Inquire)
- SYSTEM.SAMPLE.SET (dla programów przykładowych Set)

Schemat blokowy dla pliku SYSTEM.SAMPLE.ECHO ECHO jest pokazany na Rysunek 9 na stronie 1466. W tym przykładzie komenda wysyłająca do tego serwera żądanie programu RPG to:

```
CALL PGM(QMQMSAMP/AMQ3REQ4) PARM('SYSTEM.SAMPLE.ECHO
+ 30 blank characters','Queue_Manager_Name')
```

ponieważ nazwa kolejki i nazwa menedżera kolejek muszą mieć długość 48 znaków.

Uwaga: Ta przykładowa kolejka ma wyzwalacz typu FIRST, więc jeśli w kolejce znajdują się już komunikaty przed uruchomieniem przykładu żądania, aplikacje serwera nie są wyzwalane przez wysyłane komunikaty.

Jeśli chcesz spróbować dalszych przykładów, możesz wypróbować następujące warianty:

- Należy użyć komendy AMQ3TRG4 zamiast komendy AMQ3SRV4 , aby wprowadzić zadanie zamiast niego, ale potencjalne opóźnienia wprowadzenia zadania mogą spowodować, że śledzenie tego, co się dzieje, będzie mniej łatwe.
- Użyj pliku SYSTEM.SAMPLE.INQ i SYSTEM.SAMPLE.SET . Korzystając z przykładowego zbioru danych, komendy służące do wysyłania żądań programu RPG do tych serwerów to:

  ```
  CALL PGM(QMQMSAMP/AMQ3INQ4) PARM('SYSTEM.SAMPLE.INQ
  + 31 blank characters')
  CALL PGM(QMQMSAMP/AMQ3SET4) PARM('SYSTEM.SAMPLE.SET
  + 31 blank characters')
  ```

  ponieważ nazwa kolejki musi mieć długość 48 znaków.

  Te przykładowe kolejki mają również wyzwalacz typu FIRST.

Przykładowy program Echo w systemie IBM i

Przykładowe programy echo zwracają komunikat wysłany do kolejki odpowiedzi. Program ma nazwę AMQ3ECH4

Aby proces wyzwalania działał, należy upewnić się, że przykładowy program Echo, który ma być używany, jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.ECHO. W tym celu należy określić nazwę przykładowego programu Echo, który ma być używany, w polu *ApplId* definicji procesu SYSTEM.SAMPLE.ECHOPROCESS. (W tym celu można użyć komendy CHGMQMPRC, opisanej w sekcji Administrowanie produktem IBM i). Przykładowa kolejka ma wyzwalacz typu FIRST, więc jeśli w kolejce znajdują się już komunikaty przed uruchomieniem próbki żądania, próbka Echo nie jest wyzwalana przez wysyłane komunikaty.

Po poprawnym ustawieniu definicji należy najpierw uruchomić zadanie AMQ3SRV4 w jednym zadaniu, a następnie uruchomić zadanie AMQ3REQ4 w innym. Zamiast komendy AMQ3SRV4można użyć komendy AMQ3TRG4 , ale potencjalne opóźnienia wprowadzania zadań mogą ułatwić śledzenie tego, co się dzieje.

Przykładowe programy żądania służą do wysyłania komunikatów do kolejki SYSTEM.SAMPLE.ECHO. Przykładowe programy echo wysyłają komunikat odpowiedzi zawierający dane z komunikatu żądania do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Echo

Po wyzwoleniu program jawnie nawiązuje połączenie z domyślnym menedżerem kolejek przy użyciu wywołania MQCONN. Chociaż nie jest to konieczne w systemie IBM i, oznacza to, że można użyć tego samego programu na innych platformach bez zmiany kodu źródłowego.

Następnie program otwiera kolejkę o nazwie określonej w strukturze komunikatu wyzwalacza, który został przekazany podczas uruchamiania. (Dla jasności będzie to *kolejka żądań*). Program używa wywołania MQOPEN do otwarcia tej kolejki dla wejścia współużytkowanego.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. To wywołanie używa opcji GMATM i GMWT z przedziałem czasu oczekiwania wynoszącym 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania; jeśli nie, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program używa wywołania MQPUT do umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera treść komunikatu żądania.

Jeśli w kolejce żądań nie ma już żadnych komunikatów, program zamyka tę kolejkę i rozłącza się z menedżerem kolejek.

Program ten może również odpowiadać na komunikaty wysyłane do kolejki z platform innych niż IBM i, ale w takiej sytuacji nie jest dostarczana żadna próbka. Aby program ECHO działał:

- Napisz program, poprawnie określając pola *Format*, *Encoding*i *CCSID* , w celu wysyłania komunikatów żądań tekstowych.

  Program ECHO żąda od menedżera kolejek przeprowadzenia konwersji danych komunikatu, jeśli jest to konieczne.
- Podaj wartość CONVERT (*YES) w kanale nadawczym IBM MQ for IBM i , jeśli napisany program nie zapewnia podobnej konwersji dla odpowiedzi.

Przykładowy program Inquire w systemie IBM i

Przykładowy program zapytania AMQ3INQ4, za pomocą wywołania MQINQ, pyta o niektóre atrybuty kolejki.

Program jest przeznaczony do uruchamiania jako program wyzwalany, więc jego jedynym wejściem jest struktura MQTMC (komunikat wyzwalany). Ta struktura zawiera nazwę kolejki docelowej z atrybutami, których dotyczy zapytanie.

Aby proces wyzwalania działał, należy upewnić się, że program przykładowy Inquire jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.INQ. W tym celu należy podać nazwę programu przykładowego Inquire w polu *ApplId* pliku SYSTEM.SAMPLE.INQPROCESS . (W tym celu można użyć komendy CHGMQMPRC, opisanej w sekcji Zmiana procesu MQ (CHGMQMPRC)). Przykładowa kolejka ma wyzwalacz typu FIRST, więc jeśli w kolejce znajdują się już komunikaty przed uruchomieniem próbki żądania, próbka zapytania nie jest wyzwalana przez wysyłane komunikaty.

Po poprawnym ustawieniu definicji należy najpierw uruchomić zadanie AMQ3SRV4 w jednym zadaniu, a następnie uruchomić zadanie AMQ3REQ4 w innym. Zamiast komendy AMQ3SRV4można użyć komendy AMQ3TRG4 , ale potencjalne opóźnienia wprowadzania zadań mogą ułatwić śledzenie tego, co się dzieje.

Przykładowy program Request służy do wysyłania komunikatów żądań, z których każdy zawiera tylko nazwę kolejki, do kolejki SYSTEM.SAMPLE.INQ. Dla każdego komunikatu żądania program przykładowy Inquire wysyła komunikat odpowiedzi zawierający informacje o kolejce określonej w komunikacie żądania. Odpowiedzi są wysyłane do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt programu przykładowego Inquire

Po wyzwoleniu program jawnie nawiązuje połączenie z domyślnym menedżerem kolejek przy użyciu wywołania MQCONN. Chociaż nie jest to konieczne w systemie IBM i, ta opcja projektowa oznacza, że można używać tego samego programu na innych platformach bez zmiany kodu źródłowego.

Następnie program otwiera kolejkę o nazwie określonej w strukturze komunikatu wyzwalacza, który został przekazany podczas uruchamiania. (Dla jasności będzie to *kolejka żądań*). Program używa wywołania MQOPEN do otwarcia tej kolejki dla wejścia współużytkowanego.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. To wywołanie używa opcji GMATM i GMWT z przedziałem czasu oczekiwania wynoszącym 5 sekund. Program testuje

deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania; jeśli nie, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program odczytuje nazwę kolejki (która będzie zwana *kolejką docelową*). zawarte w danych i otwiera tę kolejkę za pomocą wywołania MQOPEN z opcją OOINQ. Następnie program używa wywołania MQINQ do uzyskania informacji o wartościach atrybutów **InhibitGet**, **CurrentQDepth**i **OpenInputCount** kolejki docelowej.

Jeśli wywołanie MQINQ powiedzie się, program użyje wywołania MQPUT do umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera wartości trzech atrybutów.

Jeśli wywołanie MQOPEN lub MQINQ nie powiedzie się, program użyje wywołania MQPUT do umieszczenia komunikatu *raportu* w kolejce odpowiedzi. W polu *MDFB* deskryptora komunikatu tego raportu znajduje się kod przyczyny zwrócony przez wywołanie MQOPEN lub MQINQ, w zależności od tego, które wywołanie nie powiodło się.

Po wywołaniu MQINQ program zamyka kolejkę docelową za pomocą wywołania MQCLOSE.

Jeśli w kolejce żądań nie ma już żadnych komunikatów, program zamyka tę kolejkę i rozłącza się z menedżerem kolejek.

Przykładowy program Set w systemie IBM i

Przykładowy program Set, AMQ3SET4, blokuje operacje umieszczania w kolejce za pomocą wywołania MQSET w celu zmiany atrybutu **InhibitPut** kolejki.

Program jest przeznaczony do działania jako program wyzwalany, więc jego jedynym wejściem jest struktura MQTMC (komunikat wyzwalany), która zawiera nazwę kolejki docelowej z atrybutami, do której ma zostać skierowane zapytanie.

Aby proces wyzwalania działał, należy upewnić się, że przykładowy program Set jest wyzwalany przez komunikaty przychodzące do kolejki SYSTEM.SAMPLE.SET. W tym celu należy podać nazwę przykładowego programu Set w polu *ApplId* definicji procesu SYSTEM.SAMPLE.SETPROCESS. (W tym celu można użyć komendy CHGMQMPRC, opisanej w sekcji Administrowanie produktem IBM i). Przykładowa kolejka ma wyzwalacz typu FIRST, więc jeśli w kolejce znajdują się już komunikaty przed uruchomieniem próbki żądania, próbka Ustaw nie jest wyzwalana przez wysyłane komunikaty.

Po poprawnym ustawieniu definicji należy najpierw uruchomić zadanie AMQ3SRV4 w jednym zadaniu, a następnie uruchomić zadanie AMQ3REQ4 w innym. Zamiast komendy AMQ3SRV4można użyć komendy AMQ3TRG4 , ale potencjalne opóźnienia wprowadzania zadań mogą ułatwić śledzenie tego, co się dzieje.

Przykładowy program Request służy do wysyłania komunikatów żądań, z których każdy zawiera tylko nazwę kolejki, do kolejki SYSTEM.SAMPLE.SET. Dla każdego komunikatu żądania program przykładowy Set wysyła komunikat odpowiedzi zawierający potwierdzenie, że operacje umieszczania zostały zablokowane w określonej kolejce. Odpowiedzi są wysyłane do kolejki odpowiedzi określonej w komunikacie żądania.

Projekt przykładowego programu Set

Po wyzwoleniu program jawnie nawiązuje połączenie z domyślnym menedżerem kolejek przy użyciu wywołania MQCONN. Chociaż nie jest to konieczne w systemie IBM i, oznacza to, że można użyć tego samego programu na innych platformach bez zmiany kodu źródłowego.

Następnie program otwiera kolejkę o nazwie określonej w strukturze komunikatu wyzwalacza, który został przekazany podczas uruchamiania. (Dla jasności będzie to *kolejka żądań*). Program używa wywołania MQOPEN do otwarcia tej kolejki dla wejścia współużytkowanego.

Program korzysta z wywołania MQGET w celu usunięcia komunikatów z tej kolejki. To wywołanie używa opcji GMATM i GMWT z przedziałem czasu oczekiwania wynoszącym 5 sekund. Program testuje deskryptor każdego komunikatu, aby sprawdzić, czy jest to komunikat żądania; jeśli nie, program usuwa komunikat i wyświetla komunikat ostrzegawczy.

Dla każdego komunikatu żądania usuniętego z kolejki żądań program odczytuje nazwę kolejki (która będzie zwana *kolejką docelową*). zawarte w danych i otwiera tę kolejkę za pomocą wywołania

MQOPEN z opcją OOSET. Następnie program używa wywołania MQSET do ustawienia wartości atrybutu **InhibitPut** kolejki docelowej na QAPUTI.

Jeśli wywołanie MQSET powiodło się, program używa wywołania MQPUT do umieszczenia komunikatu odpowiedzi w kolejce odpowiedzi. Ten komunikat zawiera łańcuch PUT inhibited.

Jeśli wywołanie MQOPEN lub MQSET nie powiedzie się, program użyje wywołania MQPUT do umieszczenia komunikatu *raportu* w kolejce odpowiedzi. W polu *MDFB* deskryptora komunikatu tego komunikatu raportu znajduje się kod przyczyny zwrócony przez wywołanie MQOPEN lub MQSET, w zależności od tego, które wywołanie nie powiodło się.

Po wywołaniu MQSET program zamyka kolejkę docelową za pomocą wywołania MQCLOSE.

Jeśli w kolejce żądań nie ma już żadnych komunikatów, program zamyka tę kolejkę i rozłącza się z menedżerem kolejek.

Przykładowe programy wyzwalające w systemie IBM i

IBM MQ for IBM i udostępnia dwa przykładowe programy wyzwalające napisane w języku ILE/RPG.

Są to następujące programy:

AMQ3TRG4
: Jest to monitor wyzwalacza dla środowiska IBM i . Wprowadza zadanie IBM i dla aplikacji, która ma zostać uruchomiona, ale oznacza to, że z każdym komunikatem wyzwalacza związany jest dodatkowy koszt przetwarzania.

AMQ3SRV4
: Jest to serwer wyzwalacza dla środowiska IBM i . Dla każdego komunikatu wyzwalacza ten serwer uruchamia komendę uruchamiania w swoim własnym zadaniu w celu uruchomienia określonej aplikacji. Serwer wyzwalacza może wywoływać transakcje CICS .

Wersje tych przykładów w języku C są również dostępne jako programy wykonywalne w bibliotece QMQM o nazwie AMQSTRG4 i AMQSERV4.

Przykładowy monitor wyzwalacza AMQ3TRG4 w systemie IBM i

AMQ3TRG4 jest monitorem wyzwalacza. Przyjmuje jeden parametr: nazwa kolejki inicjującej, którą ma obsługiwać. AMQSAMP4 definiuje przykładową kolejkę inicjowania, SYSTEM.SAMPLE.TRIGGER, którego można użyć podczas wypróbowania programów przykładowych.

Komenda AMQ3TRG4 wprowadza zadanie IBM i dla każdego poprawnego komunikatu wyzwalacza, który został odebrany z kolejki inicjującej.

Projekt monitora wyzwalacza

Monitor wyzwalacza otwiera kolejkę inicjującą i pobiera komunikaty z kolejki, określając nieograniczony przedział czasu oczekiwania.

Monitor wyzwalacza wprowadza zadanie IBM i w celu uruchomienia aplikacji określonej w komunikacie wyzwalacza i przekazuje strukturę MQTMC (znakowa wersja komunikatu wyzwalacza). Dane środowiska w komunikacie wyzwalacza są używane jako parametry wprowadzania zadania.

Na koniec program zamyka kolejkę inicjującą.

Przykładowy serwer wyzwalacza AMQ3SRV4

AMQ3SRV4 jest serwerem wyzwalacza. Przyjmuje jeden parametr: nazwa kolejki inicjującej, którą ma obsługiwać. AMQSAMP4 definiuje przykładową kolejkę inicjowania, SYSTEM.SAMPLE.TRIGGER, którego można użyć podczas wypróbowania programów przykładowych.

Dla każdego komunikatu wyzwalacza AMQ3SRV4 uruchamia komendę uruchamiania w swoim własnym zadaniu w celu uruchomienia określonej aplikacji.

W przypadku użycia przykładowej kolejki wyzwalacza komenda do wykonania jest następująca:

```
CALL PGM(QMQM/AMQ3SRV4) PARM('Queue Name')
```

Gdzie Queue Name musi mieć długość 48 znaków, co jest osiągane przez dopełnienie nazwy kolejki wymaganą liczbą odstępów. Oznacza to, że w przypadku korzystania z pliku SYSTEM.SAMPLE.TRIGGER jako kolejkę docelową, będzie potrzebowała 28 znaków odstępu.

Projekt serwera wyzwalaczy

Projekt serwera wyzwalaczy jest podobny do projektu monitora wyzwalaczy, z wyjątkiem serwera wyzwalaczy:

- Zezwala zarówno na aplikacje CICS , jak i IBM i
- Nie używa danych środowiska z komunikatu wyzwalacza
- Wywołuje aplikacje IBM i we własnym zadaniu (lub używa STRCICSUSR do uruchamiania aplikacji CICS) zamiast wprowadzania zadania IBM i .
- Otwiera kolejkę inicjującą dla wejścia współużytkowanego, dzięki czemu wiele serwerów wyzwalaczy może działać w tym samym czasie

Uwaga: Programy uruchomione przez komendę AMQ3SRV4 nie mogą używać wywołania MQDISC, ponieważ spowoduje to zatrzymanie serwera wyzwalaczy. Jeśli programy uruchomione przez komendę AMQ3SRV4 używają wywołania MQCONN, otrzymają kod przyczyny RC2002 .

Kończenie działania przykładowych programów wyzwalających w systemie IBM i
Program monitora wyzwalacza można zakończyć za pomocą opcji 2 żądania systemowego (ENDRQS) lub przez zablokowanie pobierania z kolejki wyzwalacza.

Jeśli używana jest przykładowa kolejka wyzwalacza, komenda jest następująca:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*NO)
```

Uwaga: Aby ponownie uruchomić wyzwalanie w tej kolejce, należy wprowadzić komendę:

```
CHGMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') GETENBL(*YES)
```

Uruchamianie przykładów przy użyciu kolejek zdalnych w systemie IBM i
Zdalne kolejkowanie można zademonstrować, uruchamiając przykłady w połączonych menedżerach kolejek komunikatów.

Program AMQSAMP4 udostępnia lokalną definicję kolejki zdalnej (SYSTEM.SAMPLE.REMOTE), który używa zdalnego menedżera kolejek o nazwie 0THER. Aby użyć tej przykładowej definicji, należy zmienić wartość 0THER na nazwę drugiego menedżera kolejek komunikatów, który ma być używany. Należy również skonfigurować kanał komunikatów między dwoma menedżerami kolejek komunikatów. Więcej informacji na ten temat zawiera sekcja Programy obsługi wyjścia kanału dla kanałów przesyłania komunikatów.

Przykładowy program żądania umieszcza własną nazwę lokalnego menedżera kolejek w polu *MDRM* komunikatów, które wysyła. Przykłady zapytań i ustawiania wysyłają komunikaty odpowiedzi do kolejki i menedżera kolejek komunikatów wymienionych w polach *MDRQ* i *MDRM* przetwarzanych komunikatów żądań.

Kody powrotu dla IBM i (ILE RPG)

Te informacje opisują kody powrotu powiązane z interfejsami MQI i MQAI.

Kody powrotu powiązane z:

- Komendy PCF (Programmable Command Format) są wymienione w sekcji Skorowidz formatów komend programowalnych.
- Wywołania C++ są wymienione w sekcji Korzystanie z języka C++.

Dla każdego wywołania menedżer kolejek lub procedura wyjścia zwraca kod zakończenia i kod przyczyny w celu wskazania powodzenia lub niepowodzenia wywołania.

Wnioski nie mogą zależeć od błędów sprawdzanych w określonej kolejności, z wyjątkiem przypadków, w których zaznaczono inaczej. Jeśli z wywołania może wynikać więcej niż jeden kod zakończenia lub kod przyczyny, zgłoszony błąd zależy od implementacji.

Kody zakończenia dla IBM i (ILE RPG)

Parametr kodu zakończenia (*CMPCOD*) umożliwia programowi wywołującemu szybkie sprawdzenie, czy wywołanie zakończyło się pomyślnie, zostało zakończone częściowo, czy nie.

CKOK
(MQCC_OK na innych platformach)

Zakończono pomyślnie.

Wywołanie zostało zakończone w pełni; wszystkie parametry wyjściowe zostały ustawione. W tym przypadku parametr **REASON** zawsze ma wartość RCNONE.

CWARN
(MQCC_WARN na innych platformach)

Ostrzeżenie (częściowe zakończenie).

Wywołanie zostało zakończone częściowo. Niektóre parametry wyjściowe mogły zostać ustawione oprócz parametrów wyjściowych *CMPCOD* i *REASON* . Parametr **REASON** udostępnia dodatkowe informacje o częściowym zakończeniu.

CCFAIL (poczta elektroniczna)
(MQCC_FAIL na innych platformach)

Wywołanie nie powiodło się.

Przetwarzanie wywołania nie zostało zakończone, a stan menedżera kolejek zwykle nie uległ zmianie. Wyjątki zostały odnotowane. Parametry wyjściowe *CMPCOD* i *REASON* zostały ustawione; inne parametry pozostają niezmienione, chyba że zaznaczono inaczej.

Przyczyną może być błąd w aplikacji lub sytuacja zewnętrzna w stosunku do programu, na przykład odebranie uprawnień użytkownika. Parametr **REASON** udostępnia dodatkowe informacje o błędzie.

Kody przyczyny dla IBM i (ILE RPG)

Parametr kodu przyczyny (*REASON*) jest kwalifikacją parametru kodu zakończenia (*CMPCOD*).

Jeśli nie ma specjalnej przyczyny, zwracana jest wartość RCNONE. Pomyślne wywołanie zwraca wartości CCOK i RCNONE.

Jeśli kodem zakończenia jest CCWARN lub CCFAIL, menedżer kolejek zawsze zgłasza kwalifikującą się przyczynę; szczegóły są podane w opisie każdego wywołania.

Tam, gdzie procedury użytkownika ustawiają kody zakończenia i przyczyny, powinny one być zgodne z tymi regułami. Ponadto wszystkie specjalne wartości przyczyn zdefiniowane przez procedury zewnętrzne powinny być mniejsze od zera, aby upewnić się, że nie powodują konfliktów z wartościami zdefiniowanymi przez menedżer kolejek. Wyjścia mogą ustawiać przyczyny, które zostały już zdefiniowane przez menedżera kolejek, tam gdzie są one odpowiednie.

Kody przyczyny występują również w:

- Pole *DLREA* struktury MQDLH
- Pole *MDFB* struktury MQMD

Pełną listę kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Aby znaleźć kod przyczyny IBM i na tej liście, należy usunąć kod "RC" z przodu, na przykład RC2002 zmieni się na 2002. Kody zakończenia są również wyświetlane w taki sam sposób, jak na innych platformach:

Tabela 814. Nazwy kodów przyczyny w systemie IBM i i na innych platformach

IBM i	**Inne platformy**
CKOK	MQCC_OK
CWARN	MQCC_WARN
CCFAIL (poczta elektroniczna)	MQCC_FAIL

Reguły sprawdzania poprawności opcji MQI dla IBM i (ILE RPG)

Ten temat zawiera informacje o sytuacjach, które generują kod przyczyny RC2046 z wywołania MQOPEN, MQPUT, MQPUT1, MQGET lub MQCLOSE.

Wywołanie MQOPEN w systemie IBM i

Opcje wywołania MQOPEN:

- *Należy określić co najmniej jedną* z następujących wartości:
  - OOBRW
  - OOINPQ,
  - OOOINPX
  - OOOINPS
  - OOINQ,
  - GOTOWE
  - OOSET,
- Dozwolony jest tylko *jeden* z następujących elementów:
  - OOINPQ,
  - OOOINPX
  - OOOINPS
- Dozwolony jest tylko *jeden* z następujących elementów:
  - OBNDO
  - OOBNDN
  - OOBNDQ

  Uwaga: Wymienione wcześniej opcje wykluczają się wzajemnie. Jednak ponieważ wartość OOBNDQ wynosi zero, podanie jej z jedną z dwóch pozostałych opcji wiązania nie powoduje wystąpienia kodu przyczyny RC2046. OOBNDQ jest dostarczane w celu wspomagania dokumentacji programu.
- Jeśli określono OOSAVA, należy również podać jedną z opcji OOINP*.
- Jeśli określono jedną z opcji OOSET* lub OOPAS*, należy również określić OOOUT.

Wywołanie MQPUT dla IBM i

Dla opcji put-message:

- Kombinacja PMSYP i PMNSYP nie jest dozwolona.
- Dozwolony jest tylko *jeden* z następujących elementów:
  - Kod PMDEFC

  - MNOC
  - PMPASA
  - PMPASI
  - PSETA
  - PSETI
- Wartość PTATU nie jest dozwolona (jest poprawna tylko w przypadku wywołania MQPUT1).

MQPUT1 wywołanie IBM i

W przypadku opcji put-message reguły są takie same, jak w przypadku wywołania MQPUT, z wyjątkiem następujących opcji:

- PMALTU jest dozwolone.
- PMLOGO nie jest dozwolone.

Wywołanie MQGET w systemie IBM i

W przypadku opcji get-message:

- Dozwolona jest tylko *jedna* z następujących opcji:
  - GMNSYP,
  - GMSYP
  - Komenda GMPSYP
- Dozwolona jest tylko *jedna* z następujących opcji:
  - GMBRWF,
  - GMBRWC
  - GMBRWN
  - GMMUC
- Komenda GMSYP nie jest dozwolona z żadną z następujących opcji:
  - GMBRWF,
  - GMBRWC
  - GMBRWN
  - GMLK,
  - GMUNLK
- Komenda GMPSYP nie jest dozwolona z żadną z następujących opcji:
  - GMBRWF,
  - GMBRWC
  - GMBRWN
  - GMCMPM,
  - GMUNLK
- Jeśli określono GMLK, należy również określić jedną z następujących opcji:
  - GMBRWF,
  - GMBRWC
  - GMBRWN
- Jeśli określono wartość GMUNLK, dozwolone są tylko następujące opcje:
  - GMNSYP,

- GMNWT,

Wywołanie MQCLOSE w systemie IBM i

- Dla opcji wywołania MQCLOSE. Kombinacja CODEL i COPURG jest niedozwolona.
- Dozwolony jest tylko jeden z następujących elementów:
  - KKPSB
  - CORMSB

Wywołanie MQSUB w systemie IBM i

W przypadku opcji wywołania MQSUB:

- Należy określić co najmniej jeden z następujących elementów:
- Należy określić co najmniej jeden z następujących elementów:
  - SOALT
  - SRES
  - SOCRT
- Dozwolony jest tylko jeden z następujących elementów:
  - SODUR
  - SONDUR

  Uwaga: Wymienione wcześniej opcje wykluczają się wzajemnie. Ponieważ jednak wartość SOnDUR wynosi zero, podanie jej z wartością SODUR nie powoduje wystąpienia kodu przyczyny RC2046. Parametr SONDUR jest udostępniany na potrzeby dokumentacji programu pomocy.
- Kombinacja SOGRP i SOMAN nie jest dozwolona.
- Komenda SOGRP wymaga określenia identyfikatora SOSCID.
- Dozwolony jest tylko jeden z następujących parametrów: SOAUID SOFUID
- Kombinacja SONEWP i SOPUBR jest niedozwolona.
- SONEWP jest dozwolony tylko w połączeniu z SOCRT.
- Dozwolony jest tylko jeden z następujących elementów:
  - SOWCHR
  - SOWTOP

Kodowania maszynowe w systemie IBM i

Te informacje umożliwiają poznanie struktury pola *MDENC* w deskryptorze komunikatu.

Więcej informacji na temat deskryptora komunikatu zawiera sekcja “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141.

Pole *MDENC* jest 32-bitową liczbą całkowitą podzieloną na cztery osobne podpola. Te podpola identyfikują:

- Kodowanie używane dla binarnych liczb całkowitych
- Kodowanie używane dla liczb całkowitych z upakowaną liczbą dziesiętną
- Kodowanie używane dla liczb zmiennopozycyjnych
- Zarezerwowane bity

Każde podpole jest identyfikowane przez maskę bitową, która ma 1 bity w pozycjach odpowiadających podpolu i 0 bitów w innym miejscu. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 najmniej znaczącym bitem. Zdefiniowane są następujące maski:

PLIMSK
Maska dla kodowania binarno-całkowitoliczbowe.

To podpole zajmuje pozycje bitów od 28 do 31 w polu *MDENC* .

ENDMSK
Maska kodowania upakowanego dziesiętnego typu integer.

To podpole zajmuje pozycje bitów od 24 do 27 w polu *MDENC* .

ENFMSK
Maska kodowania zmiennopozycyjnego.

To podpole zajmuje pozycje bitów od 20 do 23 w polu *MDENC* .

ENRMSK
Maska zarezerwowanych bitów.

To podpole zajmuje pozycje bitów od 0 do 19 w polu *MDENC* .

IBM i Kodowanie binarno-całkowite w systemie IBM i

Poprawne wartości dla kodowania binarno-całkowitego.

Następujące wartości są poprawne dla kodowania binarna-liczba całkowita:

ENIUND
Niezdefiniowane kodowanie liczb całkowitych.

Binarne liczby całkowite są reprezentowane przy użyciu niezdefiniowanego kodowania.

ENINOR
Normalne kodowanie liczb całkowitych.

Binarne liczby całkowite są reprezentowane w konwencjonalny sposób:

- Najmniej znaczący bajt w liczbie ma najwyższy adres spośród wszystkich bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres.
- Najmniej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym wyższym adresem; najbardziej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym niższym adresem.

ENIREV
Kodowanie odwrotnej liczby całkowitej.

Binarne liczby całkowite są reprezentowane w taki sam sposób, jak ENINOR, ale z bajtami ułożonymi w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak ENINOR.

IBM i Kodowanie packed-decimal-integer w systemie IBM i

Poprawne wartości dla kodowania upakowanego dziesiętnego typu integer

Następujące wartości są poprawne dla kodowania liczb dziesiętnych upakowanych liczb całkowitych:

ZAKOŃCZONE
Niezdefiniowane kodowanie z upakowaną liczbą dziesiętną.

Spakowane dziesiętne liczby całkowite są reprezentowane przy użyciu niezdefiniowanego kodowania.

ENDNOR
Normalne kodowanie z upakowaną liczbą dziesiętną.

Upakowane dziesiętne liczby całkowite są reprezentowane w konwencjonalny sposób:

- Każda cyfra dziesiętna w drukowalnej postaci liczby jest reprezentowana w upakowanej liczbie dziesiętnej przez pojedynczą cyfrę szesnastkową z zakresu od X' 0 'do X' 9'. Każda cyfra szesnastkowa zajmuje 4 bity, a każdy bajt w upakowanej liczbie dziesiętnej reprezentuje dwie cyfry dziesiętne w drukowalnej postaci liczby.

- Najmniej znaczący bajt w upakowanej liczbie dziesiętnej to bajt, który zawiera najmniej znaczącą cyfrę dziesiętną. W tym bajcie najbardziej znaczące 4 bity zawierają najmniej znaczącą cyfrę dziesiętną, a najmniej znaczące 4 bity zawierają znak. Znakiem jest X'C '(dodatni), X 'D' (ujemny) lub X'F ' (niepodpisany).
- Najmniej znaczący bajt w liczbie ma najwyższy adres spośród wszystkich bajtów w liczbie; najbardziej znaczący bajt ma najniższy adres.
- Najmniej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym wyższym adresem; najbardziej znaczący bit w każdym bajcie znajduje się obok bajtu z następnym niższym adresem.

ENDREV
Kodowanie z odwrotnym upakowanym kodowaniem dziesiętnym.

Upakowane dziesiętne liczby całkowite są reprezentowane w taki sam sposób, jak ENDNOR, ale z bajtami ułożonymi w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak ENDNOR.

IBM i Kodowanie zmiennopozycyjne w systemie IBM i

Poprawne wartości kodowania zmiennopozycyjnego

Następujące wartości są poprawne dla kodowania zmiennopozycyjnego:

ENFUND
Niezdefiniowane kodowanie zmiennopozycyjne.

Liczby zmiennopozycyjne są reprezentowane przy użyciu niezdefiniowanego kodowania.

ENFNOR
Normalne kodowanie zmiennopozycyjne IEEE (The Institute of Electrical and Electronics Engineers).

Liczby zmiennopozycyjne są reprezentowane w standardowym formacie zmiennopozycyjnym IEEE, a bajty są ułożone w następujący sposób:

- Najmniej znaczący bajt w mantysie ma najwyższy adres spośród wszystkich bajtów w liczbie; bajt zawierający wykładnik ma najniższy adres
- Najmniej znaczący bit w każdym bajcie jest obok bajtu z następnym wyższym adresem; najbardziej znaczący bit w każdym bajcie jest obok bajtu z następnym niższym adresem

Szczegóły kodowania zmiennopozycyjnego IEEE można znaleźć w standardzie IEEE 754.

ENFREV
Odwrócone kodowanie zmiennopozycyjne IEEE.

Liczby zmiennopozycyjne są reprezentowane w taki sam sposób, jak ENFNOR, ale z bajtami ułożonymi w odwrotnej kolejności. Bity w każdym bajcie są ułożone w taki sam sposób, jak ENFNOR.

ENF390
Kodowanie zmiennopozycyjne architektury System/390 .

Liczby zmiennopozycyjne są reprezentowane przez standardowy format zmiennopozycyjny System/390 ; jest on również używany przez system System/370.

IBM i Konstruowanie kodowania w systemie IBM i

Aby utworzyć wartość dla pola *MDENC* w strukturze MQMD, należy dodać odpowiednie stałe opisujące wymagane kodowania.

Należy połączyć tylko jedno kodowanie ENI* z jednym kodowaniem END* i jednym z kodowań ENF*.

IBM i Analizowanie kodowania w systemie IBM i

Pole *MDENC* zawiera podpola. W związku z tym aplikacje, które muszą sprawdzić kodowanie liczb całkowitych, upakowanych liczb dziesiętnych lub zmiennopozycyjnych, powinny korzystać z techniki opisanej w tym temacie.

Używanie arytmetyki

Poniższe kroki należy wykonać przy użyciu arytmetyki liczb całkowitych:

1. Wybierz jedną z następujących wartości, w zależności od wymaganego typu kodowania:
   - 1 dla binarnego kodowania liczb całkowitych
   - 16 dla kodowania liczb całkowitych z upakowanym kodem dziesiętnym
   - 256 dla kodowania zmiennopozycyjnego

   Wywołaj wartość A.
2. Podziel wartość pola *MDENC* przez A . Wywołaj wynik B.
3. Podziel B przez 16; wywołaj wynik C.
4. Pomnóż C przez 16 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. E jest wymaganym kodowaniem i można go przetestować pod kątem równości z każdą wartością, która jest poprawna dla tego typu kodowania.

IBM i Podsumowanie kodowania architektury komputera w systemie IBM i

Tabela podsumowująca kodowania dla architektur komputerów.

Kodowania dla architektur komputerów przedstawia Tabela 815 na stronie 1478.

Tabela 815. Podsumowanie kodowania dla architektur maszynowych

Architektura komputera	Binarne kodowanie całkowite	Kodowanie liczb całkowitych upakowanych i dziesiętnych	Kodowanie zmiennopozycyjne
IBM i	normalne	normalne	Normalny IEEE
Intel x86	Odwrotne	Odwrotne	Odwrócone IEEE
PowerPC	normalne	normalne	Normalny IEEE
System/390	normalne	normalne	System/390

IBM i Opcje raportu i flagi komunikatów w systemie IBM i

Ten temat dotyczy pól *MDREP* i *MDMFL* , które są częścią deskryptora komunikatu MQMD określonego w wywołaniach MQGET, MQPUT i MQPUT1 .

Więcej informacji na temat deskryptora komunikatu zawiera sekcja “MQMD (deskryptor komunikatu) w systemie IBM i” na stronie 1141. Informacje te opisują:

- Struktura pola raportu i sposób jego przetwarzania przez menedżer kolejek
- W jaki sposób aplikacja powinna analizować pole raportu
- Struktura pola message-flags

Struktura pola raportu

Pole *MDREP* jest 32-bitową liczbą całkowitą podzieloną na trzy osobne podpola.

Te podpola identyfikują:

- Opcje raportu, które są odrzucane, jeśli menedżer kolejek lokalnych ich nie rozpoznaje

- Opcje raportu, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek ich nie rozpoznaje
- Opcje raportu, które są akceptowane tylko wtedy, gdy spełnione są inne warunki

Każde podpole jest identyfikowane przez maskę bitową, która ma 1 bity w pozycjach odpowiadających podpolu i 0 bitów w innym miejscu. Należy zauważyć, że bity w podpolu nie zawsze są ze sobą sąsiadujące. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 najmniej znaczącym bitem. Następujące maski są zdefiniowane w celu identyfikacji pól podrzędnych:

RORUM
Maska dla odrzuconych nieobsługiwanych opcji raportu.

Ta maska identyfikuje pozycje bitowe w polu *MDREP* , w których opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, spowodują niepowodzenie wywołania MQPUT lub MQPUT1 z kodem zakończenia CCFAIL i kodem przyczyny RC2061.

To podpole zajmuje pozycje bitów 3 i od 11 do 13.

ROAUM
Maska dla nieobsługiwanych opcji raportu, które są akceptowane.

Ta maska identyfikuje pozycje bitowe w polu *MDREP* , w których opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, będą jednak akceptowane w wywołaniach MQPUT lub MQPUT1 . W tym przypadku zwracany jest kod zakończenia CCWARN z kodem przyczyny RC2104 .

To pole podrzędne zajmuje pozycje bitów od 0 do 2, od 4 do 10 i od 24 do 31.

W tym podpolu znajdują się następujące opcje raportu:

- ROCMTC,
- RODLQ,
- RODISC
- ROEXC
- ROEXCD
- ROEXCF,
- ROEXP
- ROEXPD
- ROEXPF
- RONAN
- RONMI
- BRAK
- ROPAN
- ROPCI
- ROPMI

ROAUXM
Maska dla nieobsługiwanych opcji raportu, które są akceptowane tylko w pewnych okolicznościach.

Ta maska identyfikuje pozycje bitowe w polu *MDREP* , w których opcje raportu, które nie są obsługiwane przez lokalny menedżer kolejek, będą mimo to akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem* , że spełnione są oba następujące warunki:

- Komunikat jest przeznaczony dla zdalnego menedżera kolejek.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (czyli kolejka identyfikowana przez pola *ODMN* i *ODON* w deskryptorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest lokalną kolejką transmisji).

Jeśli warunki te są spełnione, zwracany jest kod zakończenia CCWARN z kodem przyczyny RC2104 , a w przeciwnym razie kod CCFAIL z kodem przyczyny RC2061 .

To podpole zajmuje pozycje bitów od 14 do 23.

W tym podpolu znajdują się następujące opcje raportu:

- ROKOA
- ROKOD
- ROKOF
- RZT
- RODOD
- ROCODF

Jeśli w polu *MDREP* są określone opcje, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza kolejno każde podpole, używając operacji bitowej AND w celu połączenia pola *MDREP* z maską dla tego podpola. Jeśli wynik tej operacji jest różny od zera, zwracany jest kod zakończenia i opisane wcześniej kody przyczyny.

Jeśli zostanie zwrócona wartość CCWARN, nie jest ona zdefiniowana, który kod przyczyny jest zwracany w przypadku wystąpienia innych warunków ostrzegawczych.

Możliwość określenia i zaakceptowania opcji raportu, które nie są rozpoznawane przez lokalny menedżer kolejek, jest przydatna, gdy konieczne jest wysłanie komunikatu z opcją raportu, która zostanie rozpoznana i przetworzona przez *zdalny* menedżer kolejek.

IBM i Analizowanie pola raportu w systemie IBM i

Pole MDREP zawiera podpola. Z tego powodu niektóre aplikacje muszą sprawdzić, czy nadawca komunikatu zażądał konkretnego raportu. Aplikacje te powinny korzystać z techniki opisanej w tym temacie.

Używanie arytmetyki

Poniższe kroki należy wykonać przy użyciu arytmetyki liczb całkowitych:

1. Wybierz jedną z następujących wartości, w zależności od typu raportu, który ma zostać sprawdzony:
   - ROCOA dla raportu COA
   - Raport ROCOD dla COD
   - ROEXC dla raportu o wyjątkach
   - ROEXP dla raportu o utracie ważności

   Wywołaj wartość A.
2. Podziel pole *MDREP* przez A . Wywołaj wynik B.
3. Podziel B przez 8 ; Wywołaj wynik C.
4. Pomnóż C przez 8 i odejmij od B ; Wywołaj wynik D.
5. Pomnóż D przez A ; Wywołaj wynik E.
6. Przetestuj E pod kątem równości z każdą z wartości, które są możliwe dla tego typu raportu.

   Na przykład, jeśli A to ROEXC, należy przetestować E pod kątem równości z każdym z poniższych kryteriów, aby określić, co zostało określone przez nadawcę komunikatu:

   - BRAK
   - ROEXC
   - ROEXCD
   - ROEXCF,

   Testy mogą być wykonywane w dowolnej kolejności, która jest najwygodniejsza dla logiki aplikacji.

Poniższa pseudocode ilustruje tę technikę dla komunikatów raportu o wyjątkach:

```
A = ROEXC
B = Report/A
C = B/8
D = B - C*8
E = D*A
```

Podobnej metody można użyć do przetestowania opcji ROPMI lub ROPCI; należy wybrać wartość A , która z tych dwóch stałych jest odpowiednia, a następnie postępować zgodnie z wcześniejszym opisem, zastępując wartość 8 w poprzednich krokach wartością 2.

IBM i Struktura pola flagi komunikatu w systemie IBM i

Pole *MDMFL* jest 32-bitową liczbą całkowitą podzieloną na trzy osobne podpola.

Te podpola identyfikują:

- Flagi komunikatów, które są odrzucane, jeśli lokalny menedżer kolejek ich nie rozpoznaje
- Flagi komunikatów, które są zawsze akceptowane, nawet jeśli lokalny menedżer kolejek ich nie rozpoznaje
- Flagi komunikatów, które są akceptowane tylko wtedy, gdy spełnione są inne warunki

Uwaga: Wszystkie podpola w programie *MDMFL* są zarezerwowane do użytku przez menedżer kolejek.

Każde podpole jest identyfikowane przez maskę bitową, która ma 1 bity w pozycjach odpowiadających podpolu i 0 bitów w innym miejscu. Bity są numerowane w taki sposób, że bit 0 jest najbardziej znaczącym bitem, a bit 31 najmniej znaczącym bitem. Następujące maski są zdefiniowane w celu identyfikacji pól podrzędnych:

MFRUM
Maska dla odrzuconych nieobsługiwanych flag komunikatów.

Ta maska identyfikuje pozycje bitowe w polu *MDMFL* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, spowodują niepowodzenie wywołania MQPUT lub MQPUT1 z kodem zakończenia CCFAIL i kodem przyczyny RC2249.

To podpole zajmuje pozycje bitów od 20 do 31.

W tym polu podrzędnym znajdują się następujące flagi komunikatów:

- MFLMIG@ item: indo
- MFLSEG
- MMIG
- MSEG
- MSEGA
- MSEGI

MFAUM
Maska akceptowanych nieobsługiwanych flag komunikatów.

Ta maska identyfikuje pozycje bitowe w polu *MDMFL* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, będą jednak akceptowane w wywołaniach MQPUT lub MQPUT1 . Kod zakończenia to CCOK.

To podpole zajmuje pozycje bitów od 0 do 11.

MFAUXM
Maska dla nieobsługiwanych flag komunikatów, które są akceptowane tylko w pewnych okolicznościach.

Ta maska identyfikuje pozycje bitowe w polu *MDMFL* , w których flagi komunikatów, które nie są obsługiwane przez lokalny menedżer kolejek, będą mimo to akceptowane w wywołaniach MQPUT lub MQPUT1 *pod warunkiem* , że spełnione są oba następujące warunki:

- Komunikat jest przeznaczony dla zdalnego menedżera kolejek.
- Aplikacja nie umieszcza komunikatu bezpośrednio w lokalnej kolejce transmisji (czyli kolejka identyfikowana przez pola *ODMN* i *ODON* w deskryptorze obiektu określonym w wywołaniu MQOPEN lub MQPUT1 nie jest lokalną kolejką transmisji).

Jeśli te warunki są spełnione, zwracany jest kod zakończenia CCOK, a jeśli nie, CCFAIL z kodem przyczyny RC2249 .

To podpole zajmuje pozycje bitów od 12 do 19.

Jeśli w polu *MDMFL* określono flagi, których menedżer kolejek nie rozpoznaje, menedżer kolejek sprawdza kolejno każde podpole, używając operacji bitowej AND w celu połączenia pola *MDMFL* z maską dla tego podpola. Jeśli wynik tej operacji jest różny od zera, zwracany jest kod zakończenia i opisane wcześniej kody przyczyny.

IBM i Konwersja danych w systemie IBM i

W tym temacie opisano interfejs do wyjścia konwersji danych oraz przetwarzanie wykonywane przez menedżer kolejek, gdy wymagana jest konwersja danych.

Wyjście konwersji danych jest wywoływane w ramach przetwarzania wywołania MQGET. Jest on używany do przekształcania danych komunikatu aplikacji w reprezentację wymaganą przez aplikację odbierającą. Konwersja danych komunikatu aplikacji jest opcjonalna i wymaga określenia opcji GMCONV w wywołaniu MQGET.

Poniżej opisano następujące aspekty konwersji danych:

- Przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję GMCONV; patrz sekcja “Przetwarzanie konwersji w systemie IBM i” na stronie 1482.
- Konwencje przetwarzania używane przez menedżer kolejek podczas przetwarzania wbudowanego formatu. Konwencje te są zalecane również w przypadku programów zewnętrznych napisanych przez użytkownika. Patrz sekcja “Konwencje przetwarzania w systemie IBM i” na stronie 1484.
- Specjalne uwagi dotyczące konwersji komunikatów raportu; patrz sekcja “Konwersja komunikatów raportów w systemie IBM i” na stronie 1487.
- Parametry przekazywane do wyjścia konwersji danych; patrz sekcja “MQCONVX (wyjście konwersji danych) w systemie IBM i” na stronie 1498.
- Wywołanie, którego można użyć z wyjścia w celu przekształcenia danych znakowych między różnymi reprezentacjami; patrz sekcja “MQXCNVC (Konwersja znaków) w systemie IBM i” na stronie 1493.
- Parametr struktury danych, który jest specyficzny dla wyjścia; patrz sekcja “MQDXP (parametr wyjścia konwersji danych) w systemie IBM i” na stronie 1488.

IBM i Przetwarzanie konwersji w systemie IBM i

Te informacje opisują przetwarzanie wykonywane przez menedżer kolejek w odpowiedzi na opcję GMCONV.

Menedżer kolejek wykonuje następujące działania, jeśli w wywołaniu MQGET podano opcję GMCONV i istnieje komunikat, który ma zostać zwrócony do aplikacji:

1. Jeśli spełniony jest co najmniej jeden z poniższych warunków, konwersja nie jest konieczna:
   - Dane komunikatu są już w zestawie znaków i kodowaniu wymaganym przez aplikację wywołującą wywołanie MQGET. Przed wykonaniem wywołania aplikacja musi ustawić pola *MDCSI* i *MDENC* w parametrze **MSGDSC** wywołania MQGET na wymagane wartości.
   - Długość danych komunikatu wynosi zero.
   - Długość parametru **BUFFER** wywołania MQGET wynosi zero.

   W takich przypadkach komunikat jest zwracany bez konwersji do aplikacji, która wywołała wywołanie MQGET. Wartości *MDCSI* i *MDENC* w parametrze **MSGDSC** są ustawiane na wartości w informacjach

sterujących komunikatu, a wywołanie kończy się jedną z następujących kombinacji kodu zakończenia i kodu przyczyny:

Kod zakończenia
Kod przyczyny

CKOK
BRAK RCNONE

CWARN
RC2079

CWARN
RC2080

Poniższe kroki są wykonywane tylko wtedy, gdy zestaw znaków lub kodowanie danych komunikatu różni się od odpowiedniej wartości parametru **MSGDSC** i istnieją dane do przekształcenia:

1. Jeśli pole *MDFMT* w informacjach sterujących komunikatu ma wartość FMNONE, komunikat jest zwracany bez konwersji z kodem zakończenia CCWARN i kodem przyczyny RC2110.

   We wszystkich innych przypadkach przetwarzanie konwersji jest kontynuowane.
2. Komunikat zostanie usunięty z kolejki i umieszczony w tymczasowym buforze, którego wielkość jest taka sama jak wielkość parametru **BUFFER** . W przypadku operacji przeglądania komunikat jest kopiowany do buforu tymczasowego, a nie usuwany z kolejki.
3. Jeśli komunikat musi zostać obcięty, aby zmieścił się w buforze, wykonywane są następujące czynności:
   - Jeśli opcja GMATM nie została podana, komunikat jest zwracany bez konwersji z kodem zakończenia CCWARN i kodem przyczyny RC2080.
   - Jeśli określono opcję *był* GMATM, kod zakończenia jest ustawiany na CCWARN, kod przyczyny jest ustawiany na RC2079, a przetwarzanie konwersji jest kontynuowane.
4. Jeśli komunikat może być zapisany w buforze bez obcięcia lub podano opcję GMATM, wykonywane są następujące czynności:
   - Jeśli format jest wbudowany, bufor jest przekazywany do usługi konwersji danych menedżera kolejek.
   - Jeśli format nie jest formatem wbudowanym, bufor jest przekazywany do zapisanej przez użytkownika procedury zewnętrznej, która ma taką samą nazwę jak format. Jeśli nie można znaleźć wyjścia, komunikat jest zwracany bez konwersji z kodem zakończenia CCWARN i kodem przyczyny RC2110.

   Jeśli nie wystąpi żaden błąd, dane wyjściowe z usługi konwersji danych lub z wyjścia napisanego przez użytkownika są przekształcanym komunikatem oraz kodem zakończenia i kodem przyczyny, który ma zostać zwrócony do aplikacji wywołującej wywołanie MQGET.
5. Jeśli konwersja powiedzie się, menedżer kolejek zwraca przekształcony komunikat do aplikacji. W takim przypadku kod zakończenia i kod przyczyny zwracane przez wywołanie MQGET będą zwykle jedną z następujących kombinacji:

   Kod zakończenia
   Kod przyczyny

   CKOK
   BRAK RCNONE

   CWARN
   RC2079

   Jeśli jednak konwersja jest wykonywana przez program zewnętrzny napisany przez użytkownika, mogą zostać zwrócone inne kody przyczyny, nawet jeśli konwersja zakończyła się pomyślnie.

   Jeśli konwersja nie powiedzie się (z dowolnej przyczyny), menedżer kolejek zwraca nieprzekształcony komunikat do aplikacji z polami *MDCSI* i *MDENC* w parametrze **MSGDSC** ustawionymi na wartości w informacjach sterujących komunikatu oraz kodem zakończenia CCWARN.

IBM i Konwencje przetwarzania w systemie IBM i

Podczas przekształcania formatu wbudowanego menedżer kolejek jest zgodny z konwencjami przetwarzania opisanymi w tym temacie.

Należy rozważyć zastosowanie tych konwencji do programów zewnętrznych napisanych przez użytkownika, chociaż nie jest to wymuszane przez menedżer kolejek. Formaty wbudowane przekształcane przez menedżer kolejek są następujące:

- FMADMN
- FMMDE,
- FMCICKI
- FMPCF,
- FMCMD1
- FMRMH
- FMCMD2
- FMRFH
- FMDLH
- FMRFH2
- FMDH
- FMSTR,
- FMEVNT,
- FMTM,
- FMIMS
- FMXQH
- FMIMVS,

1. Jeśli komunikat jest rozwijany podczas konwersji i przekracza wielkość parametru **BUFFER** , wykonywane są następujące czynności:
   - Jeśli opcja GMATM nie została podana, komunikat jest zwracany bez konwersji z kodem zakończenia CCWARN i kodem przyczyny RC2120.
   - Jeśli opcja GMATM *została* określona, komunikat jest obcinany, kod zakończenia jest ustawiany na CCWARN, kod przyczyny jest ustawiany na RC2079, a przetwarzanie konwersji jest kontynuowane.
2. Jeśli wystąpi obcięcie (przed lub podczas konwersji), liczba poprawnych bajtów zwracanych w parametrze **BUFFER** może być *mniejsza niż* długość buforu.

   Może to wystąpić na przykład wtedy, gdy 4-bajtowa liczba całkowita lub znak DBCS znajduje się na końcu buforu. Niekompletny element informacji nie jest przekształcany, dlatego bajty w zwróconym komunikacie nie zawierają poprawnych informacji. Może to również wystąpić, jeśli komunikat, który został obcięty przed konwersją, zostanie zmniejszony podczas konwersji.

   Jeśli liczba zwróconych poprawnych bajtów jest mniejsza niż długość buforu, nieużywane bajty na końcu buforu są ustawiane na wartości null.
3. Jeśli tablica lub łańcuch znajduje się na końcu buforu, konwertowana jest jak najwięcej danych; tylko określony element tablicy lub znak DBCS, który jest niekompletny, nie jest konwertowany- konwertowane są poprzedzające go elementy tablicy lub znaki.
4. Jeśli wystąpi obcięcie (przed lub podczas konwersji), długość zwracana dla parametru **DATLEN** jest długością komunikatu *unconverted* przed obcięcia.
5. Jeśli łańcuchy są konwertowane między zestawami znaków jednobajtowych (SBCS), zestawami znaków dwubajtowych (DBCS) lub zestawami znaków wielobajtowych (MBCS), łańcuchy mogą się rozszerzać lub zwijać.

- W formatach PCF FMADMN, FMEVNT i FMPCF łańcuchy w strukturach MQCFST i MQCFSL są odpowiednio rozwijane lub zwiane w celu dostosowania do łańcucha po konwersji.

  W przypadku struktury listy łańcuchów MQCFSL łańcuchy na liście mogą być rozwijane lub zwiane o różne kwoty. W takim przypadku menedżer kolejek dopełni krótsze łańcuchy odstępami, aby były one takie same jak najdłuższy łańcuch po konwersji.
- W formacie FMRMH łańcuchy adresowane przez pola RMSEO, RMSNO, RMDEOi RMDNO rozszerzają się lub kurczają zgodnie z potrzebami w celu dostosowania łańcuchów po konwersji.
- W formacie FMRFH pole RFNVS jest rozwijane lub zwijane zgodnie z potrzebami w celu uwzględnienia par nazwa-wartość po konwersji.
- W strukturach o stałej wielkości pól menedżer kolejek zezwala na rozwijanie lub zwijania łańcuchów w obrębie pól stałych, jeśli nie utracono istotnych informacji. W związku z tym końcowe odstępy i znaki występujące po pierwszym znaku o kodzie zero w polu są traktowane jako nieistotne.
  - Jeśli łańcuch zostanie rozwinięty, ale tylko nieistotne znaki muszą zostać usunięte, aby pomieścić przekształcony łańcuch w polu, konwersja powiedzie się, a wywołanie zostanie zakończone z kodem CCOK i kodem przyczyny RCNONE (przy założeniu, że nie ma innych błędów).
  - Jeśli łańcuch zostanie rozwinięty, ale przekształcony łańcuch wymaga usunięcia znaczących znaków, aby zmieścił się w polu, komunikat zostanie zwrócony bez konwersji, a wywołanie zostanie zakończone z kodem CCWARN i kodem przyczyny RC2190.

    Uwaga: Kod przyczyny RC2190 powoduje w tym przypadku określenie opcji GMATM.
  - Jeśli łańcuch się kontrakty, menedżer kolejek dopełnić łańcuch odstępami do długości pola.

6. W przypadku komunikatów składających się z jednej lub większej liczby struktur nagłówka IBM MQ , po których następują dane użytkownika, możliwe jest przekształcenie jednej lub większej liczby struktur nagłówka, podczas gdy pozostała część komunikatu nie jest przekształcana. Jednak z dwoma wyjątkami pola MDCSI i MDENC w każdej strukturze nagłówka zawsze poprawnie wskazują zestaw znaków i kodowanie danych, które następują po strukturze nagłówka.

   Dwa wyjątki to struktury MQCIH i MQIIH, w których wartości w polach MDCSI i MDENC w tych strukturach nie są istotne. W przypadku tych struktur dane znajdujące się po strukturze mają ten sam zestaw znaków i kodowanie, co struktura MQCIH lub MQIIH.

7. Jeśli pola MDCSI lub MDENC w informacjach sterujących pobieranego komunikatu lub w parametrze **MSGDSC** określają wartości, które nie są zdefiniowane lub nie są obsługiwane, menedżer kolejek może zignorować błąd, jeśli niezdefiniowana lub nieobsługiwana wartość nie musi być używana podczas przekształcania komunikatu.

   Na przykład, jeśli w polu MDENC w komunikacie określono nieobsługiwane kodowanie zmiennopozycyjne, ale komunikat zawiera tylko dane całkowite lub zawiera dane zmiennopozycyjne, które nie wymagają konwersji (ponieważ źródłowe i docelowe kodowanie zmiennopozycyjne są identyczne), błąd może, ale nie musi zostać rozpoznany.

   Jeśli błąd został zdiagnozowany, komunikat jest zwracany bez konwersji, z kodem zakończenia CCWARN i jednym z kodów przyczyny RC2111, RC2112, RC2113, RC2114 lub RC2115, RC2116, RC2117, RC2118 (w zależności od przypadku); pola MDCSI i MDENC w parametrze **MSGDSC** są ustawione na wartości podane w informacjach sterujących w komunikacie.

   Jeśli błąd nie zostanie zdiagnozowany i konwersja zakończy się pomyślnie, wartości zwracane w polach MDCSI i MDENC w parametrze **MSGDSC** to wartości określone przez aplikację wywołującą wywołanie MQGET.

8. We wszystkich przypadkach, jeśli komunikat jest zwracany do aplikacji, która nie została przekształcona, kod zakończenia jest ustawiany na CCWARN, a pola MDCSI i MDENC w parametrze **MSGDSC** są ustawiane na wartości odpowiednie dla nieprzekształconych danych. Jest to również wykonywane dla FMNONE.

   Parametr **REASON** jest ustawiony na kod, który wskazuje, dlaczego nie można przeprowadzić konwersji, chyba że komunikat również musiał zostać obcięty. Kody przyczyny związane z obcięciem mają pierwszeństwo przed kodami przyczyny związanymi z konwersją. (Aby określić, czy obcięty

komunikat został przekształcony, sprawdź wartości zwrócone w polach MDCSI i MDENC w parametrze **MSGDSC**).

Po zdiagnozowaniu błędu zwracany jest konkretny kod przyczyny lub ogólny kod przyczyny RC2119. Zwrócony kod przyczyny zależy od możliwości diagnostycznych bazowej usługi konwersji danych.

9. W przypadku zwrócenia kodu zakończenia CCWARN i zastosowania więcej niż jednego kodu przyczyny kolejność wykonywania jest następująca:
   a. Następująca przyczyna ma pierwszeństwo przed wszystkimi innymi:
      - RC2079
   b. Następny w kolejności jest następujący powód:
      - RC2110
   c. Kolejność wykonywania operacji w pozostałych kodach przyczyny nie jest zdefiniowana.
10. Po zakończeniu wywołania MQGET:
    - Następujący kod przyczyny wskazuje, że komunikat został pomyślnie przekształcony:
      – BRAK RCNONE
    - Następujący kod przyczyny wskazuje, że komunikat *mógł* zostać pomyślnie przekształcony (sprawdź zawartość pól MDCSI i MDENC w parametrze **MSGDSC** , aby uzyskać informacje):
      – RC2079
    - Wszystkie inne kody przyczyny wskazują, że komunikat nie został przekształcony.

Następujące przetwarzanie jest specyficzne dla formatów wbudowanych; nie dotyczy formatów zdefiniowanych przez użytkownika:

1. Z wyjątkiem następujących formatów:
   - FMADMN
   - FMEVNT,
   - FMIMVS,
   - FMPCF,
   - FMSTR,

   żaden z wbudowanych formatów nie może być przekształcony z lub na zestawy znaków, które nie zawierają znaków SBCS dla znaków, które są poprawne w nazwach kolejek. Jeśli zostanie podjęta próba przeprowadzenia takiej konwersji, komunikat zostanie zwrócony bez konwersji z kodem zakończenia CCWARN i kodem przyczyny RC2111 lub RC2115.

   Zestaw znaków Unicode UTF-16 jest przykładem zestawu znaków, który nie zawiera znaków SBCS dla znaków, które są poprawne w nazwach kolejek.
2. Jeśli dane komunikatu dla wbudowanego formatu są obcięte, pola w komunikacie, które zawierają długości łańcuchów lub liczby elementów lub struktur, nie są dopasowywane w celu odzwierciedlenia długości danych zwracanych do aplikacji. Wartości zwracane dla takich pól w danych komunikatu są wartościami mającymi zastosowanie do komunikatu przed obcięciem.

   Podczas przetwarzania komunikatów, takich jak obcięty komunikat FMADMN, należy zachować ostrożność, aby aplikacja nie próbowała uzyskać dostępu do danych poza końcem zwracanych danych.
3. Jeśli nazwa formatu to FMDLH, dane komunikatu rozpoczynają się od struktury MQDLH, po której może wystąpić zero lub więcej bajtów danych komunikatu aplikacji. Format, zestaw znaków i kodowanie danych komunikatu aplikacji są definiowane przez pola DLFMT, DLCSIi DLENC w strukturze MQDLH na początku komunikatu. Ponieważ struktura MQDLH i dane komunikatu aplikacji mogą mieć różne zestawy znaków i kodowania, istnieje możliwość, że jedna, druga lub obie struktury MQDLH i dane komunikatu aplikacji wymagają konwersji.

   W razie potrzeby menedżer kolejek najpierw przekształca strukturę MQDLH. Jeśli konwersja powiedzie się lub struktura MQDLH nie wymaga konwersji, menedżer kolejek sprawdza pola DLCSI i DLENC w strukturze MQDLH, aby sprawdzić, czy konwersja danych komunikatu aplikacji jest wymagana. Jeśli

konwersja jest wymagana, menedżer kolejek wywołuje zapisane przez użytkownika wyjście z nazwą nadaną przez pole DLFMT w strukturze MQDLH lub wykonuje samą konwersję (jeśli DLFMT jest nazwą wbudowanego formatu).

Jeśli wywołanie MQGET zwraca kod zakończenia CCWARN, a kod przyczyny wskazuje, że konwersja nie powiodła się, ma zastosowanie jedna z następujących sytuacji:

- Nie można przekształcić struktury MQDLH. W takim przypadku dane komunikatu aplikacji nie zostaną również przekształcone.
- Struktura MQDLH została przekształcona, ale dane komunikatu aplikacji nie zostały przekształcone.

Aplikacja może sprawdzić wartości zwrócone w polach MDCSI i MDENC w parametrze **MSGDSC** oraz w strukturze MQDLH w celu określenia, które z powyższych wartości mają zastosowanie.

4. Jeśli nazwa formatu to FMXQH, dane komunikatu rozpoczynają się od struktury MQXQH, po której może wystąpić zero lub więcej bajtów dodatkowych danych. Te dodatkowe dane są zwykle danymi komunikatu aplikacji (które mogą mieć zerową długość), ale na początku dodatkowych danych może być również obecna jedna lub więcej dodatkowych struktur nagłówka IBM MQ .

   Struktura MQXQH musi być w zestawie znaków i kodowaniu menedżera kolejek. Format, zestaw znaków i kodowanie danych następujących po strukturze MQXQH są określone przez pola MDFMT, MDCSIi MDENC w strukturze MQMD zawartej w MQXQH. Dla każdej obecnej struktury nagłówka IBM MQ pola MDFMT, MDCSIi MDENC w strukturze opisują dane, które następują po tej strukturze; dane te są albo inną strukturą nagłówka IBM MQ , albo danymi komunikatu aplikacji.

   Jeśli opcja GMCONV jest określona dla komunikatu FMXQH, dane komunikatu aplikacji i niektóre struktury nagłówka MQ są konwertowane, ale dane w strukturze MQXQH nie są konwertowane. W przypadku powrotu z wywołania MQGET:

   - Wartości pól MDFMT, MDCSIi MDENC w parametrze **MSGDSC** opisują dane w strukturze MQXQH, a nie dane komunikatu aplikacji. Z tego powodu wartości nie będą takie same jak wartości określone przez aplikację, która wywołała wywołanie MQGET.

     W rezultacie aplikacja, która wielokrotnie pobiera komunikaty z kolejki transmisji z określoną opcją GMCONV, musi zresetować pola MDCSI i MDENC w parametrze **MSGDSC** do wartości niezbędnych dla danych komunikatu aplikacji przed każdym wywołaniem MQGET.
   - Wartości pól MDFMT, MDCSIi MDENC w ostatniej strukturze nagłówka produktu MQ opisują dane komunikatu aplikacji. Jeśli nie ma innych struktur nagłówka IBM MQ , dane komunikatu aplikacji są opisane przez te pola w strukturze MQMD w strukturze MQXQH. Jeśli konwersja powiedzie się, wartości będą takie same jak wartości określone w parametrze **MSGDSC** przez aplikację, która wywołała wywołanie MQGET.

   Jeśli komunikat jest komunikatem listy dystrybucyjnej, po strukturze MQXQH następuje struktura MQDH (plus jej tablice rekordów MQOR i MQPMR), po której może następować zero lub więcej dalszych struktur nagłówka IBM MQ i zero lub więcej bajtów danych komunikatu aplikacji. Podobnie jak struktura MQXQH, struktura MQDH musi być w zestawie znaków i kodowaniu menedżera kolejek i nie jest przekształcana w wywołaniu MQGET, nawet jeśli określono opcję GMCONV.

   Opisane wcześniej przetwarzanie struktur MQXQH i MQDH jest przeznaczone przede wszystkim do użycia przez agenty kanału komunikatów podczas pobierania komunikatów z kolejek transmisji.

IBM i Konwersja komunikatów raportów w systemie IBM i

Komunikat raportu może zawierać różne ilości danych komunikatu aplikacji, w zależności od opcji raportu określonych przez nadawcę oryginalnego komunikatu.

W szczególności komunikat raportu może zawierać:

1. Brak danych komunikatu aplikacji
2. Niektóre dane komunikatu aplikacji z oryginalnego komunikatu

   Dzieje się tak, gdy nadawca oryginalnego komunikatu określa RO* D, a komunikat jest dłuższy niż 100 bajtów.

3. Wszystkie dane komunikatu aplikacji z oryginalnego komunikatu

   Dzieje się tak, gdy nadawca oryginalnego komunikatu określa RO* F lub RO* D, a komunikat jest 100 bajtów lub krótszy.

Gdy menedżer kolejek lub agent kanału komunikatów generuje komunikat raportu, kopiuje nazwę formatu z oryginalnego komunikatu do pola *MDFMT* w informacjach sterujących w komunikacie raportu. Nazwa formatu w komunikacie raportu może zatem sugerować długość danych, która różni się od długości w komunikacie raportu (przypadki 1 i 2 opisane wcześniej).

Jeśli podczas pobierania komunikatu raportu podano opcję GMCONV:

- W opisanym wcześniej przypadku 1 wyjście konwersji danych nie zostanie wywołane (ponieważ komunikat raportu nie będzie zawierać danych).
- W opisanym wcześniej przypadku 3 nazwa formatu poprawnie określa długość danych komunikatu.
- Jednak w przypadku 2 opisanym wcześniej, wyjście konwersji danych zostanie wywołane w celu przekształcenia komunikatu, który jest *krótszy* niż długość określona przez nazwę formatu.

  Ponadto kodem przyczyny przekazywanym do wyjścia będzie zazwyczaj RCNONE (kod przyczyny nie oznacza, że komunikat został obcięty). Dzieje się tak, ponieważ dane komunikatu zostały obcięte przez *nadawcę* komunikatu raportu, a nie przez menedżer kolejek odbiorcy w odpowiedzi na wywołanie MQGET.

Ze względu na te możliwości wyjście konwersji danych nie powinno używać nazwy formatu do określenia długości przekazywanych do niego danych; zamiast tego wyjście powinno sprawdzać długość dostarczonych danych i być przygotowane do konwersji mniejszej ilości danych niż długość określona przez nazwę formatu. Jeśli dane mogą zostać pomyślnie przekształcone, kod zakończenia CCOK i kod przyczyny RCNONE powinny zostać zwrócone przez wyjście. Długość danych komunikatu do przekształcenia jest przekazywana do wyjścia jako parametr **INLEN** .

Interfejs programistyczny wrażliwy na produkt

Jeśli komunikat raportu zawiera informacje o działaniu, które miało miejsce, jest on nazywany raportem działania. Przykładami działań są:

- Agent MCA wysyłający komunikat z kolejki w dół kanału
- Agent MCA odbierający komunikat z kanału i umieszczający go w kolejce
- Niedostarczony komunikat MCA umieszczający w kolejce komunikat, którego nie można dostarczyć.
- Agent MCA pobierający komunikat z kolejki i odrzucający go
- Program do obsługi niedostarczonych komunikatów umieszczający komunikat z powrotem w kolejce
- serwer komend przetwarzający żądanie PCF-broker przetwarzający żądanie publikowania
- Aplikacja użytkownika, która otrzymuje komunikat z kolejki-aplikacja użytkownika, która przegląda komunikat w kolejce

Każda aplikacja, w tym menedżer kolejek, może dodać część danych komunikatu do raportu działań po nagłówku raportu. Ilość danych, które powinny być dostarczone, jeśli niektóre są wysyłane, nie jest ustalona i jest ona ustalana przez aplikację. Zwracane informacje powinny być przydatne dla aplikacji przetwarzającej raport działań. Raporty działań menedżera kolejek zwrócą wraz z nimi wszystkie standardowe struktury nagłówków IBM MQ (rozpoczynające się od nagłówka MQH) zawarte w oryginalnym komunikacie. Obejmuje to na przykład wszystkie nagłówki MQRFH2 , które zostały dołączone do oryginalnego komunikatu. Ponadto menedżer kolejek zwróci znaleziony nagłówek MQCFH, ale nie powiązane z nim parametry PCF. Dzięki temu aplikacje monitorujące będą mogły dowiedzieć się, o co chodziło w komunikacie.

IBM i MQDXP (parametr wyjścia konwersji danych) w systemie IBM i

Blok parametru wyjścia konwersji danych.

Przegląd

Cel: Struktura MQDXP jest parametrem, który jest przekazywany przez menedżer kolejek do wyjścia konwersji danych podczas wywoływania wyjścia w celu przekształcenia danych komunikatu w ramach przetwarzania wywołania MQGET. Szczegółowe informacje na temat wyjścia konwersji danych można znaleźć w opisie wywołania MQCONVX.

Zestaw znaków i kodowanie: dane znakowe w MQDXP znajdują się w zestawie znaków lokalnego menedżera kolejek. Jest to określone przez atrybut menedżera kolejek systemu **CodedCharSetId** . Dane liczbowe w MQDXP są kodowane przy użyciu rodzimego kodowania maszynowego, które jest nadawane przez ENNAT.

Składnia: tylko pola *DXLEN*, *DXCC*, *DXREA* i *DXRES* w MQDXP mogą zostać zmienione przez wyjście. Zmiany w innych polach są ignorowane. Pola *DXLEN* nie można jednak zmienić, jeśli przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego.

Gdy sterowanie powraca do menedżera kolejek z wyjścia, menedżer kolejek sprawdza wartości zwrócone w MQDXP. Jeśli zwrócone wartości są niepoprawne, menedżer kolejek kontynuuje przetwarzanie tak, jakby wyjście zwróciło kod XRFAIL w produkcie *DXRES* . Jednak menedżer kolejek ignoruje wartości pól *DXCC* i *DXREA* zwrócone przez wyjście w tym przypadku i używa zamiast nich wartości, które miały w *danych wejściowych* do wyjścia. Następujące wartości w MQDXP powodują takie przetwarzanie:

- Pole *DXRES* nie jest typu XROK ani XRFAIL
- Pole *DXCC* nie jest typu CCOK ani CCWARN
- *DXLEN* pole mniejsze niż zero lub *DXLEN* pole zmienione, gdy przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego.

- "Pola" na stronie 1489
- "Deklaracja RPG (kopiowanie pliku CMQDXPH)" na stronie 1493

Pola

Struktura MQDXP zawiera następujące pola. Pola są opisane w **porządku alfabetycznym**:

DXAOP (10-cyfrowa liczba całkowita ze znakiem)

Opcje aplikacji.

Jest to kopia pola *GMOPT* struktury MQGMO określonej przez aplikację wywołującą wywołanie MQGET. Wyjście może wymagać sprawdzenia tych danych w celu określenia, czy podano opcję GMATM.

Jest to pole wejściowe do wyjścia.

DXCC (10-cyfrowa liczba całkowita ze znakiem)

Kod zakończenia.

Po wywołaniu wyjścia zawiera ono kod zakończenia, który zostanie zwrócony do aplikacji, która wywołała wywołanie MQGET, jeśli wyjście nie podejmie żadnych działań. Jest to zawsze CCWARN, ponieważ komunikat został obcięty lub komunikat wymaga konwersji i nie zostało to jeszcze wykonane.

W przypadku danych wyjściowych z wyjścia to pole zawiera kod zakończenia, który ma zostać zwrócony do aplikacji w parametrze **CMPCOD** wywołania MQGET. Poprawne są tylko wartości CCOK i CCWARN. Opis pola *DXREA* zawiera sugestie dotyczące sposobu, w jaki wyjście powinno ustawić to pole w danych wyjściowych.

Jest to pole wejściowe/wyjściowe do wyjścia.

DXCSI (10-cyfrowa liczba całkowita ze znakiem)

Zestaw znaków wymagany przez aplikację.

Jest to identyfikator kodowanego zestawu znaków dla zestawu znaków wymaganego przez aplikację, która wywołała wywołanie MQGET. Więcej szczegółów zawiera pole *MDCSI* w strukturze MQMD. Jeśli

aplikacja określa wartość specjalną CSQM w wywołaniu MQGET, menedżer kolejek zmienia ją na rzeczywisty identyfikator zestawu znaków używanego przez menedżer kolejek przed wywołaniem wyjścia.

Jeśli konwersja powiedzie się, wyjście powinno skopiować to do pola *MDCSI* w deskryptorze komunikatu.

Jest to pole wejściowe do wyjścia.

DXENC (10-cyfrowa liczba całkowita ze znakiem)

Kodowanie liczbowe wymagane przez aplikację.

Jest to kodowanie liczbowe wymagane przez aplikację wydającą wywołanie MQGET. Więcej szczegółów zawiera pole *MDENC* w strukturze MQMD.

Jeśli konwersja powiedzie się, wyjście powinno skopiować to do pola *MDENC* w deskryptorze komunikatu.

Jest to pole wejściowe do wyjścia.

DXHCN (10-cyfrowa liczba całkowita ze znakiem)

Uchwyt połączenia.

Jest to uchwyt połączenia, którego można użyć w wywołaniu MQXCNVC. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

DXLEN (10-cyfrowa liczba całkowita ze znakiem)

Długość danych komunikatu w bajtach.

Po wywołaniu wyjścia to pole zawiera oryginalną długość danych komunikatu aplikacji. Jeśli komunikat został obcięty, aby zmieścił się w buforze udostępnionym przez aplikację, wielkość komunikatu dostarczanego do wyjścia będzie *mniejsza* niż wartość parametru *DXLEN*. Wielkość komunikatu przekazanego do wyjścia jest zawsze określana przez parametr **INLEN** wyjścia, bez względu na ewentualne obcięcie.

Obcięcie jest wskazywane przez pole *DXREA* o wartości RC2079 na wejściu do wyjścia.

Większość konwersji nie wymaga zmiany tej długości, ale w razie potrzeby może to zrobić wyjście. Wartość ustawiona przez wyjście jest zwracana do aplikacji w parametrze **DATLEN** wywołania MQGET. Nie można jednak zmienić tej długości, jeśli przekształcany komunikat jest segmentem, który zawiera tylko część komunikatu logicznego. Wynika to z faktu, że zmiana długości spowodowałaby, że przesunięcia późniejszych segmentów w komunikacie logicznym byłyby niepoprawne.

Należy zauważyć, że jeśli wyjście ma zmienić długość danych, należy pamiętać, że menedżer kolejek już zdecydował, czy dane komunikatu będą pasować do buforu aplikacji na podstawie długości danych *nieprzekształconych* . Ta decyzja określa, czy komunikat jest usuwany z kolejki (lub przenoszony jest kursor przeglądania dla żądania przeglądania) i nie ma na niego wpływu żadna zmiana długości danych spowodowana przez konwersję. Z tego powodu zaleca się, aby wyjścia konwersji nie powodowały zmiany długości danych komunikatu aplikacji.

Jeśli konwersja znaków oznacza zmianę długości, łańcuch może zostać przekształcony w inny łańcuch o takiej samej długości w bajtach, obcinając końcowe odstępy lub dopełniając odstępami, jeśli jest to konieczne.

Wyjście nie jest wywoływane, jeśli komunikat nie zawiera danych komunikatu aplikacji, dlatego wartość *DXLEN* jest zawsze większa od zera.

Jest to pole wejściowe/wyjściowe do wyjścia.

DXREA (10-cyfrowa liczba całkowita ze znakiem)

Kod przyczyny określający *DXCC*.

Po wywołaniu wyjścia zawiera ono kod przyczyny, który zostanie zwrócony do aplikacji, która wywołała wywołanie MQGET, jeśli wyjście nie podejmie żadnych działań. Możliwe wartości to RC2079, co

oznacza, że komunikat został obcięty w celu dopasowania do buforu udostępnionego przez aplikację, oraz RC2119, co oznacza, że komunikat wymaga konwersji, ale nie został jeszcze wykonane.

W przypadku danych wyjściowych z wyjścia to pole zawiera przyczynę, która ma zostać zwrócona do aplikacji w parametrze **REASON** wywołania MQGET. Zalecane jest wykonanie następujących czynności:

- Jeśli parametr *DXREA* ma wartość RC2079 na wejściu do wyjścia, pola *DXREA* i *DXCC* nie powinny być zmieniane bez względu na to, czy konwersja powiedzie się, czy nie.

  Jeśli pole *DXCC* nie jest polem CCOK, aplikacja pobierająca komunikat może zidentyfikować niepowodzenie konwersji, porównując zwrócone wartości *MDENC* i *MDCSI* w deskryptorze komunikatu z żądanymi wartościami. Natomiast aplikacja nie może odróżnić obciętego komunikatu od komunikatu, który właśnie dopasowano do buforu. Z tego powodu kod RC2079 powinien zostać zwrócony zamiast przyczyny, która wskazuje niepowodzenie konwersji.
- Jeśli parametr *DXREA* miał inną wartość na wejściu do wyjścia:
  - Jeśli konwersja powiedzie się, parametr *DXCC* należy ustawić na wartość CCOK, a parametr *DXREA* na wartość RCNONE.
  - Jeśli konwersja nie powiedzie się lub komunikat zostanie rozwinięty i musi zostać obcięty, aby zmieścić się w buforze, należy ustawić parametr *DXCC* na CCWARN (lub pozostawić bez zmian), a parametr *DXREA* na jedną z wartości i na poniższej liście, aby wskazać rodzaj niepowodzenia.

    Należy zauważyć, że jeśli komunikat po konwersji jest zbyt duży dla buforu, powinien zostać obcięty tylko wtedy, gdy aplikacja, która wywołała wywołanie MQGET, określiła opcję GMATM:

    - Jeśli podano tę opcję, powinna zostać zwrócona przyczyna RC2079 .
    - Jeśli nie podano tej opcji, komunikat powinien zostać zwrócony bez konwersji z kodem przyczyny RC2120.

Kody przyczyny z poniższej listy są zalecane do użycia przez wyjście w celu wskazania przyczyny niepowodzenia konwersji, ale wyjście może zwrócić inne wartości z zestawu kodów RC*, jeśli uzna to za stosowne. Ponadto zakres wartości od RC0900 do RC0999 jest przydzielany do użycia przez wyjście w celu wskazania warunków, które mają być używane przez wyjście do komunikacji z aplikacją wywołującą wywołanie MQGET.

Uwaga: Jeśli nie można pomyślnie przekształcić komunikatu, wyjście musi zwrócić wartość XRFAIL w polu *DXRES* , aby menedżer kolejek zwrócił nieprzekształcony komunikat. Dzieje się tak niezależnie od kodu przyczyny zwróconego w polu *DXREA* .

RC0900
(900, X'384 ') Najmniejsza wartość dla kodu przyczyny zdefiniowanego przez aplikację.

RC0999
(999, X'3E7') Najwyższa wartość dla kodu przyczyny zdefiniowanego przez aplikację.

RC2120
(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

RC2119
(2119, X'847 ') Dane komunikatu nie zostały przekształcone.

RC2111
(2111, X'83F') Niepoprawny identyfikator źródłowego kodowanego zestawu znaków.

RC2113
(2113, X'841 ') Nie rozpoznano kodowania dziesiętnego w komunikacie.

RC2114
(2114, X'842 ') Nierozpoznane kodowanie zmiennopozycyjne w komunikacie.

RC2112
(2112, X'840 ') Nierozpoznane kodowanie źródłowe liczby całkowitej.

RC2115
(2115, X'843 ') Niepoprawny identyfikator kodowanego zestawu znaków.

RC2117
(2117, X'845 ') Nierozpoznane kodowanie Packed-decimal określone przez odbiornik.

RC2118
(2118, X'846 ') Nierozpoznane kodowanie zmiennopozycyjne określone przez odbiornik.

RC2116
(2116, X'844 ') Nie rozpoznano kodowania docelowej liczby całkowitej.

RC2079
(2079, X'81F') Zwrócono obcięty komunikat (przetwarzanie zakończone).

Jest to pole wejściowe/wyjściowe do wyjścia.

DXRES (10-cyfrowa liczba całkowita ze znakiem)

Odpowiedź z wyjścia.

Jest ona ustawiana przez wyjście, aby wskazać powodzenie lub brak konwersji. Musi to być jedna z następujących wartości:

KROK
Konwersja powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca do aplikacji, która wywołała wywołanie MQGET, następujące dane:

- Wartość pola *DXCC* na wyjściu z wyjścia
- Wartość pola *DXREA* na wyjściu z wyjścia
- Wartość pola *DXLEN* na wyjściu z wyjścia
- Zawartość buforu wyjściowego wyjścia *OUTBUF*. Liczba zwróconych bajtów jest mniejszą z wartości parametru **OUTLEN** wyjścia oraz wartość pola *DXLEN* w danych wyjściowych z wyjścia

Jeśli pola *MDENC* i *MDCSI* w parametrze deskryptora komunikatu wyjścia mają *oba* niezmienione wartości, menedżer kolejek zwraca:

- Wartość pól *MDENC* i *MDCSI* w strukturze MQDXP na *wejściu* do wyjścia

Jeśli jedno lub oba pola *MDENC* i *MDCSI* w parametrze deskryptora komunikatu wyjścia zostały zmienione, menedżer kolejek zwraca:

- Wartości pól *MDENC* i *MDCSI* w parametrze deskryptora komunikatu wyjścia w danych wyjściowych wyjścia
-

XRFAIL,
Konwersja nie powiodła się.

Jeśli wyjście określa tę wartość, menedżer kolejek zwraca do aplikacji, która wywołała wywołanie MQGET, następujące dane:

- Wartość pola *DXCC* na wyjściu z wyjścia
- Wartość pola *DXREA* na wyjściu z wyjścia
- Wartość pola *DXLEN* w polu *input* dla wyjścia.
- Zawartość buforu wejściowego wyjścia *INBUF*. Liczba zwracanych bajtów jest określona przez parametr **INLEN** .

Jeśli wyjście zmieniło wartość *INBUF*, wyniki są niezdefiniowane.

DXRES jest polem wyjściowym z wyjścia.

DXSID (4-bajtowy łańcuch znaków)

Identyfikator struktury.

Wartość musi być następująca:

DXSIDV
Identyfikator struktury parametru wyjścia konwersji danych.

Jest to pole wejściowe do wyjścia.

DXVER (10-cyfrowa liczba całkowita ze znakiem)

Numer wersji struktury.

Wartość musi być następująca:

DXVER1
Numer wersji struktury parametru wyjścia konwersji danych.

Następująca stała określa numer wersji bieżącej:

DXVERC
Bieżąca wersja struktury parametru wyjścia konwersji danych.

Uwaga: Po wprowadzeniu nowej wersji tej struktury układ istniejącej części nie ulega zmianie. Dlatego wyjście powinno sprawdzić, czy wartość w polu *DXVER* jest równa lub większa od najniższej wersji, która zawiera pola, które mają być używane przez wyjście.

Jest to pole wejściowe do wyjścia.

DXXOP (10-cyfrowa liczba całkowita ze znakiem)

Zarezerwowane.

Jest to pole zastrzeżone, a jego wartością jest 0.

Deklaracja RPG (kopiowanie pliku CMQDXPH)

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQDXP Structure
D*
D* Structure identifier
D  DXSID                  1      4
D* Structure version number
D  DXVER                  5      8I 0
D* Reserved
D  DXXOP                  9     12I 0
D* Application options
D  DXAOP                 13     16I 0
D* Numeric encoding required by application
D  DXENC                 17     20I 0
D* Character set required by application
D  DXCSI                 21     24I 0
D* Length in bytes of message data
D  DXLEN                 25     28I 0
D* Completion code
D  DXCC                  29     32I 0
D* Reason code qualifying DXCC
D  DXREA                 33     36I 0
D* Response from exit
D  DXRES                 37     40I 0
D* Connection handle
D  DXHCN                 41     44I 0
```

IBM i MQXCNVC (Konwersja znaków) w systemie IBM i

Wywołanie MQXCNVC przekształca znaki z jednego zestawu znaków w inny.

To wywołanie jest częścią interfejsu DCI (IBM MQ Data Conversion Interface), który jest jednym z interfejsów środowiska IBM MQ . Uwaga: tego wywołania można użyć tylko z wyjścia konwersji danych.

- “Składnia” na stronie 1494
- “Parametry” na stronie 1494

- "Wywołanie RPG (ILE)" na stronie 1498

Składnia

```
MQXCNVC (HCONN, OPTS, SRCCSI, SRCLEN, SRCBUF, TGTCSI, TGTLEN,
   TGTBUF, DATLEN, CMPCOD, REASON)
```

Parametry

Wywołanie MQXCNVC ma następujące parametry:

HCONN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Zwykle powinien to być uchwyt przekazywany do wyjścia konwersji danych w polu DXHCN struktury MQDXP. Ten uchwyt nie musi być taki sam, jak uchwyt określony przez aplikację, która wywołała wywołanie MQGET.

W systemie IBM idla parametru HCONNmożna podać następującą wartość specjalną:

HHCDEFH
Domyślny uchwyt połączenia.

OPTS (10-cyfrowa liczba całkowita ze znakiem)-wejście

Opcje sterujące działaniem MQXCNVC.

Można określić dowolną liczbę opcji opisanych w dalszej części tej sekcji. Jeśli wymagana jest więcej niż jedna, wartości mogą zostać dodane (nie należy dodawać tej samej stałej więcej niż raz).

Domyślna-opcja konwersji: Poniższa opcja steruje użyciem domyślnej konwersji znaków:

DCCDEF,
Konwersja domyślna.

Ta opcja określa, że domyślna konwersja znaków może być używana, jeśli jeden lub oba zestawy znaków określone w wywołaniu nie są obsługiwane. Umożliwia to menedżerowi kolejek użycie podczas konwersji łańcucha domyślnego zestawu znaków określonego przez instalację, który jest zbliżony do określonego zestawu znaków.

Uwaga: Wynikiem użycia przybliżonego zestawu znaków do konwersji łańcucha jest to, że niektóre znaki mogą być przekształcane niepoprawnie. Można tego uniknąć, używając w łańcuchu tylko tych znaków, które są wspólne zarówno dla określonego zestawu znaków, jak i dla domyślnego zestawu znaków.

Domyślne zestawy znaków są definiowane przez opcję konfiguracji podczas instalowania lub restartowania menedżera kolejek.

Jeśli nie określono DCCDEF, menedżer kolejek używa tylko określonych zestawów znaków do konwersji łańcucha i wywołanie kończy się niepowodzeniem, jeśli jeden lub oba zestawy znaków nie są obsługiwane.

Opcja dopełniania: Poniższa opcja umożliwia menedżerowi kolejek dopełnienie przekształconego łańcucha odstępami lub usunięcie nieistotnych znaków końcowych w celu dopasowania przekształconego łańcucha do buforu docelowego:

DCCFIL
Wypełnij bufor docelowy.

Ta opcja wymaga, aby konwersja była wykonywana w taki sposób, aby bufor docelowy był całkowicie zapełniony:

- Jeśli podczas konwersji łańcuch jest przekształcany, dodawane są końcowe odstępy w celu wypełnienia buforu docelowego.

- Jeśli łańcuch jest rozwijany podczas konwersji, znaki końcowe, które nie są istotne, są odrzucane, aby przekształcony łańcuch pasował do buforu docelowego. Jeśli można to zrobić pomyślnie, wywołanie kończy się z kodem CCOK i kodem przyczyny RCNONE.

  Jeśli na końcu jest zbyt mało nieistotnych znaków, w buforze docelowym jest umieszczany łańcuch, który zostanie dopasowany, a wywołanie zostanie zakończone z kodem CCWARN i kodem przyczyny RC2120.

  Nieistotne znaki to:

  – Odstępy końcowe
  – Znaki występujące po pierwszym znaku o kodzie zero w łańcuchu (z wyłączeniem samego pierwszego znaku o kodzie zero)
- Jeśli w łańcuchu TGTCSI i TGTLEN nie można całkowicie ustawić buforu docelowego przy użyciu poprawnych znaków, wywołanie kończy się niepowodzeniem z kodem CCFAIL i kodem przyczyny RC2144. Taka sytuacja może wystąpić, gdy TGTCSI jest czystym zestawem znaków DBCS (na przykład UTF-16), ale TGTLEN określa nieparzystą liczbę bajtów.
- Wartość TGTLEN może być mniejsza lub większa niż SRCLEN. W przypadku powrotu z MQXCNVC DATLEN ma taką samą wartość jak TGTLEN.

Jeśli ta opcja nie jest określona:

- Łańcuch może zwiać lub rozszerzać się w obrębie buforu docelowego zgodnie z wymaganiami. Nieistotne znaki końcowe nie są dodawane ani odrzucane.

  Jeśli przekształcony łańcuch pasuje do buforu docelowego, wywołanie zostanie zakończone z kodem CCOK i kodem przyczyny RCNONE.

  Jeśli przekształcony łańcuch jest zbyt duży dla buforu docelowego, w buforze docelowym zostanie umieszczona część łańcucha, która zostanie dopasowana, a wywołanie zostanie zakończone z kodem CCWARN i kodem przyczyny RC2120. Należy zauważyć, że w tym przypadku może zostać zwrócona mniej niż *TGTLEN* bajtów.
- Wartość TGTLEN może być mniejsza lub większa niż SRCLEN. W przypadku powrotu z MQXCNVC wartość DATLEN jest mniejsza lub równa TGTLEN.

Opcje kodowania: Następujące opcje mogą być używane do określenia kodowania liczb całkowitych łańcuchów źródłowych i docelowych. Odpowiednie kodowanie jest używane tylko wtedy, gdy odpowiedni identyfikator zestawu znaków wskazuje, że reprezentacja zestawu znaków w pamięci głównej jest zależna od kodowania używanego dla binarnych liczb całkowitych. Dotyczy to tylko niektórych zestawów znaków wielobajtowych (na przykład zestawów znaków UTF-16).

Kodowanie jest ignorowane, jeśli zestaw znaków jest jednobajtowym zestawem znaków (SBCS) lub wielobajtowym zestawem znaków z reprezentacją w pamięci głównej, która nie jest zależna od kodowania liczb całkowitych.

Należy podać tylko jedną z wartości DCCS*, w połączeniu z jedną z wartości DCCT*:

DCCSNA
Kodowanie źródłowe jest domyślnym kodowaniem dla środowiska i języka programowania.

Numer DCCSNO
Kodowanie źródłowe jest normalne.

DCCSRE,
Kodowanie źródłowe zostało odwrócone.

DCCSUN
Kodowanie źródłowe jest niezdefiniowane.

DCCTNA
Kodowanie docelowe jest domyślne dla środowiska i języka programowania.

NUMER DCCT
Kodowanie docelowe jest normalne.

DCCTRE
Kodowanie docelowe jest odwrócone.

DCCTUN
Kodowanie docelowe jest niezdefiniowane.

Zdefiniowane wcześniej wartości kodowania można dodać bezpośrednio do pola OPTS . Jeśli jednak kodowanie źródłowe lub docelowe jest uzyskiwane z pola MDENC w strukturze MQMD lub innej strukturze, należy wykonać następujące przetwarzanie:

1. Kodowanie na podstawie liczb całkowitych musi zostać wyodrębnione z pola MDENC przez wyeliminowanie kodowania liczb zmiennopozycyjnych i upakowanych liczb dziesiętnych. Szczegółowe informacje na ten temat zawiera sekcja "Analizowanie kodowania w systemie IBM i" na stronie 1477 .
2. Kodowanie liczb całkowitych wynikające z kroku 1 musi zostać pomnożone przez odpowiedni współczynnik przed dodaniem do pola OPTS . Czynniki te są następujące:

   DCCSFA,
   Współczynnik kodowania źródłowego

   DCCTFA
   Współczynnik kodowania docelowego

Jeśli ta opcja nie zostanie określona, opcje kodowania będą domyślnie ustawione na wartość undefined (DCC* UN). W większości przypadków nie ma to wpływu na pomyślne zakończenie wywołania MQXCNVC. Jeśli jednak odpowiedni zestaw znaków jest zestawem znaków wielobajtowych z reprezentacją zależną od kodowania (na przykład zestaw znaków UTF-16), wywołanie nie powiedzie się z kodem przyczyny RC2112 lub RC2116 .

Opcja domyślna: Jeśli nie określono żadnej z opisanych wcześniej opcji, można użyć następującej opcji:

DCCNON
Nie określono żadnych opcji.

DCCNON jest zdefiniowany w celu wspomagania dokumentacji programu. Ta opcja nie jest przeznaczona do użycia z żadną inną opcją, ale ponieważ jej wartość wynosi zero, takie użycie nie może zostać wykryte.

SRCCSI (10-cyfrowa liczba całkowita ze znakiem)-dane wejściowe

Identyfikator kodowanego zestawu znaków łańcucha przed konwersją.

Jest to identyfikator kodowanego zestawu znaków łańcucha wejściowego w pliku SRCBUF.

SRCLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość łańcucha przed konwersją.

Jest to długość łańcucha wejściowego (w bajtach) w SRCBUF ; musi mieć wartość zero lub większą.

SRCBUF (1-bajtowy łańcuch znaków x SRCLEN)-wejście

Łańcuch do przekształcenia.

Jest to bufor zawierający łańcuch, który ma zostać przekształcony z jednego zestawu znaków na inny.

TGTCSI (10-cyfrowa liczba całkowita ze znakiem)-wejście

Identyfikator kodowanego zestawu znaków łańcucha po konwersji.

Jest to identyfikator kodowanego zestawu znaków, na który ma zostać przekształcony kod SRCBUF .

TGTLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość buforu wyjściowego.

Jest to długość buforu wyjściowego w bajtach TGTBUF ; musi mieć wartość zero lub większą. Może być mniejsza lub większa niż SRCLEN.

TGTBUF (1-bajtowy łańcuch znaków x TGTLEN)-wyjście

Łańcuch po konwersji.

Jest to łańcuch po przekształceniu go w zestaw znaków zdefiniowany przez TGTCSI. Przekształcony łańcuch może być krótszy lub dłuższy niż nieprzekształcony łańcuch. Parametr **DATLEN** wskazuje liczbę zwróconych poprawnych bajtów.

DATLEN (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Długość łańcucha wyjściowego.

Jest to długość łańcucha zwracanego w buforze wyjściowym TGTBUF. Przekształcony łańcuch może być krótszy lub dłuższy niż nieprzekształcony łańcuch.

CMPCOD (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod zakończenia.

Jest to jedna z poniższych nazw:

CKOK
Zakończono pomyślnie.

CWARN
Ostrzeżenie (częściowe zakończenie).

CCFAIL (poczta elektroniczna)
Wywołanie nie powiodło się.

PRZYCZYNA (10-cyfrowa liczba całkowita ze znakiem)-dane wyjściowe

Kod przyczyny określający CMPCOD.

Jeśli CMPCOD to CCOK:

BRAK RCNONE
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli CMPCOD ma wartość CCWARN:

RC2120
(2120, X'848 ') Przekształcone dane są zbyt duże dla buforu.

Jeśli CMPCOD to CCFAIL:

RC2010
(2010, X'7DA') Niepoprawny parametr długości danych.

RC2150
(2150, X'866 ') Niepoprawny łańcuch DBCS.

RC2018
(2018, X'7E2') Uchwyt połączenia jest niepoprawny.

RC2046
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

RC2102
(2102, X'836 ') Niewystarczające zasoby systemowe.

RC2145
(2145, X'861 ') Niepoprawny parametr buforu źródłowego.

RC2111
(2111, X'83F') Niepoprawny identyfikator źródłowego kodowanego zestawu znaków.

RC2112
(2112, X'840 ') Nierozpoznane kodowanie źródłowe liczby całkowitej.

RC2143
(2143, X'85F') Niepoprawny parametr długości źródła.

RC2071
(2071, X'817 ') Niewystarczająca ilość dostępnej pamięci.

RC2146
(2146, X'862 ') Niepoprawny parametr buforu docelowego.

RC2115
(2115, X'843 ') Niepoprawny identyfikator kodowanego zestawu znaków.

RC2116
(2116, X'844 ') Nie rozpoznano kodowania docelowej liczby całkowitej.

RC2144
(2144, X'860 ') Niepoprawny parametr długości celu.

RC2195
(2195, X'893 ') Wystąpił nieoczekiwany błąd.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja "Kody powrotu dla IBM i (ILE RPG)" na stronie 1471.

Wywołanie RPG (ILE)

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     MQXCNVC(HCONN : OPTS : SRCCSI :
C                                       SRCLEN : SRCBUF : TGTCSI :
C                                       TGTLEN : TGTBUF : DATLEN :
C                                       CMPCOD : REASON)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
DMQXCNVC          PR                  EXTPROC('MQXCNVC')
D* Connection handle
D HCONN                         10I 0 VALUE
D* Options that control the action of MQXCNVC
D OPTS                          10I 0 VALUE
D* Coded character set identifier of string before conversion
D SRCCSI                        10I 0 VALUE
D* Length of string before conversion
D SRCLEN                        10I 0 VALUE
D* String to be converted
D SRCBUF                          *   VALUE
D* Coded character set identifier of string after conversion
D TGTCSI                        10I 0 VALUE
D* Length of output buffer
D TGTLEN                        10I 0 VALUE
D* String after conversion
D TGTBUF                          *   VALUE
D* Length of output string
D DATLEN                        10I 0
D* Completion code
D CMPCOD                        10I 0
D* Reason code qualifying CMPCOD
D REASON                        10I 0
```

IBM i MQCONVX (wyjście konwersji danych) w systemie IBM i

Ta definicja wywołania opisuje parametry, które są przekazywane do wyjścia konwersji danych.

Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQCONVX (patrz uwaga dotycząca użycia "11" na stronie 1500).

Ta definicja jest częścią interfejsu DCI (IBM MQ Data Conversion Interface), który jest jednym z interfejsów środowiska IBM MQ .

- "Składnia" na stronie 1499
- "Użycie notatek" na stronie 1499

- “Parametry” na stronie 1500
- “Wywołanie RPG (ILE)” na stronie 1502

Składnia

(MQDXP, MQMD, INLEN, INBUF, OUTLEN, OUTBUF) **MQCONVX**

Użycie notatek

1. Wyjście konwersji danych jest zapisanym przez użytkownika wyjściem, które odbiera sterowanie podczas przetwarzania wywołania MQGET. Funkcja wykonywana przez wyjście konwersji danych jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami opisanymi w tym miejscu i w powiązanej strukturze parametrów MQDXP.

   Języki programowania, które mogą być używane dla wyjścia konwersji danych, są określane przez środowisko.
2. Wyjście jest wywoływane tylko wtedy, gdy *wszystkie* następujące instrukcje są prawdziwe:
   - Opcja GMCONV jest określona w wywołaniu MQGET
   - Pole *MDFMT* w deskryptorze komunikatu nie ma wartości FMNONE.
   - Komunikat nie znajduje się jeszcze w wymaganej reprezentacji, czyli jeden lub oba elementy *MDCSI* i *MDENC* komunikatu różnią się od wartości określonej przez aplikację w deskryptorze komunikatu dostarczonym w wywołaniu MQGET.
   - Konwersja menedżera kolejek nie została jeszcze zakończona pomyślnie
   - Długość buforu aplikacji jest większa od zera
   - Długość danych komunikatu jest większa od zera
   - Kod przyczyny używany do tej pory podczas operacji MQGET to RCNONE lub RC2079
3. Podczas zapisywania wyjścia należy zwrócić uwagę na kodowanie wyjścia w taki sposób, aby umożliwić mu konwersję obciętych komunikatów. Obcięte komunikaty mogą wystąpić w następujący sposób:
   - Aplikacja odbierająca udostępnia bufor, który jest mniejszy niż komunikat, ale określa opcję GMATM w wywołaniu MQGET.

     W tym przypadku pole *DXREA* w parametrze **MQDXP** na wejściu do wyjścia będzie miało wartość RC2079.
   - Nadawca komunikatu obciął go przed wysłaniem. Taka sytuacja może wystąpić w przypadku komunikatów raportu, na przykład (więcej szczegółów zawiera sekcja “Konwersja komunikatów raportów w systemie IBM i” na stronie 1487).

     W tym przypadku pole *DXREA* w parametrze **MQDXP** na wejściu do wyjścia będzie miało wartość RCNONE (jeśli aplikacja odbierająca udostępniła bufor, który był wystarczająco duży dla komunikatu).

   Dlatego wartość pola *DXREA* na wejściu do wyjścia nie może być zawsze używana do określenia, czy komunikat został obcięty.

   Cechą wyróżniającą obciętego komunikatu jest to, że długość podana dla wyjścia w parametrze **INLEN** będzie *mniejsza* niż długość określona przez nazwę formatu zawartą w polu *MDFMT* deskryptora komunikatu. Dlatego wyjście powinno sprawdzić wartość *INLEN* przed podjęciem próby przekształcenia jakichkolwiek danych. Wyjście *nie powinno* zakładać, że podano pełną ilość danych implikowanych przez nazwę formatu.

   Jeśli wyjście nie zostało zapisane w celu przekształcenia obciętych komunikatów, a wartość **INLEN** jest mniejsza niż oczekiwana, wyjście powinno zwrócić wartość XRFAIL w polu *DXRES* parametru **MQDXP** , z polem *DXCC* ustawionym na CCWARN i polem *DXREA* ustawionym na RC2110.

   Jeśli wyjście *zostało* zapisane w celu przekształcenia obciętych komunikatów, wyjście powinno przekształcić jak najwięcej danych (patrz uwaga dotycząca następnego użycia), uważając, aby

nie podejmować próby sprawdzenia lub przekształcenia danych poza końcem pliku *INBUF*. Jeśli konwersja zakończy się pomyślnie, wyjście powinno pozostawić pole *DXREA* w parametrze **MQDXP** bez zmian. Zwraca wartość RC2079 , jeśli komunikat został obcięty przez menedżera kolejek odbiornika, lub wartość RCNONE, jeśli komunikat został obcięty przez nadawcę komunikatu.

Możliwe jest również rozwinięcie komunikatu *podczas konwersji* do miejsca, w którym jest on większy niż *OUTBUF*. W takim przypadku wyjście musi zdecydować, czy komunikat ma zostać obcięty; pole *DXAOP* w parametrze **MQDXP** będzie wskazywać, czy aplikacja odbierająca określiła opcję GMATM.

4. Ogólnie zaleca się, aby wszystkie dane w komunikacie udostępnionym do wyjścia w produkcie *INBUF* były przekształcane lub aby żaden z nich nie był przekształcany. Wyjątek od tej zasady występuje jednak wtedy, gdy komunikat jest obcinany przed konwersją lub podczas konwersji; w takim przypadku na końcu buforu może znajdować się niekompletny element (na przykład: jeden bajt znaku dwubajtowego lub 3 bajty 4-bajtowej liczby całkowitej). W takiej sytuacji zaleca się pominięcie niekompletnego elementu i ustawienie nieużywanych bajtów w pliku *OUTBUF* na wartość null. Jednak kompletne elementy lub znaki w tablicy lub łańcuchu *powinny* zostać przekształcone.
5. Jeśli wyjście jest potrzebne po raz pierwszy, menedżer kolejek próbuje załadować obiekt o takiej samej nazwie jak format (oprócz rozszerzeń). Załadowany obiekt musi zawierać wyjście przetwarzające komunikaty o tej nazwie formatu. Zaleca się, aby nazwa wyjścia i nazwa obiektu, który zawiera wyjście, były identyczne, chociaż nie wszystkie środowiska tego wymagają.
6. Nowa kopia wyjścia jest ładowana, gdy aplikacja próbuje pobrać pierwszy komunikat, który używa tego komunikatu *MDFMT* od czasu połączenia aplikacji z menedżerem kolejek. Nowa kopia może zostać również załadowana w innym czasie, jeśli menedżer kolejek odrzucił wcześniej załadowaną kopię. Z tego powodu wyjście nie powinno próbować używać pamięci statycznej do przekazywania informacji z jednego wywołania wyjścia do następnego-wyjście może zostać rozładowane między dwoma wywołaniami.
7. Jeśli istnieje wyjście dostarczone przez użytkownika o takiej samej nazwie jak jeden z wbudowanych formatów obsługiwanych przez menedżer kolejek, wyjście dostarczone przez użytkownika nie zastępuje wbudowanej procedury konwersji. Jedynymi okolicznościami, w których takie wyjście jest wywoływane, są:
   - Jeśli wbudowana procedura konwersji nie może obsłużyć konwersji do lub z systemu *MDCSI* lub *MDENC* , lub
   - Jeśli nie powiodła się konwersja danych przez wbudowaną procedurę konwersji (na przykład z powodu pola lub znaku, którego nie można przekształcić).
8. Zasięg wyjścia jest zależny od środowiska. Aby zminimalizować ryzyko konfliktów z innymi formatami, należy wybrać nazwy *MDFMT* . Zaleca się, aby nazwy te rozpoczynały się od znaków, które identyfikują aplikację definiującą nazwę formatu.
9. Wyjście konwersji danych działa w środowisku podobnym do środowiska programu, który wywołał wywołanie MQGET; środowisko zawiera przestrzeń adresową i profil użytkownika (jeśli ma zastosowanie). Program może być agentem kanału komunikatów wysyłającym komunikaty do docelowego menedżera kolejek, który nie obsługuje konwersji komunikatów. Wyjście nie może naruszyć integralności menedżera kolejek, ponieważ nie działa w środowisku menedżera kolejek.
10. Jedynym wywołaniem MQI, które może być używane przez wyjście, jest MQXCNVC. Próba użycia innych wywołań MQI nie powiodła się z kodem przyczyny RC2219lub innymi nieprzewidywalnymi błędami.
11. Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQCONVX. Nazwa wyjścia powinna być taka sama jak nazwa formatu (nazwa zawarta w polu *MDFMT* w strukturze MQMD), chociaż nie jest to wymagane we wszystkich środowiskach.

Parametry

Wywołanie MQCONVX ma następujące parametry:

MQDXP (MQDXP)-wejście/wyjście

Blok parametru wyjścia konwersji danych.

Ta struktura zawiera informacje dotyczące wywołania wyjścia. Wyjście ustawia informacje w tej strukturze w celu wskazania wyniku konwersji. Szczegółowe informacje na temat pól w tej strukturze zawiera sekcja “MQDXP (parametr wyjścia konwersji danych) w systemie IBM i” na stronie 1488 .

MQMD (MQMD)-wejście/wyjście

Deskryptor komunikatu.

Na wejściu do wyjścia jest to deskryptor komunikatu, który zostałby zwrócony do aplikacji, gdyby nie przeprowadzono konwersji. Dlatego zawiera on elementy *MDFMT*, *MDENC*i *MDCSI* nieprzekształconego komunikatu zawartego w pliku *INBUF*.

Uwaga: Parametr **MQMD** przekazywany do wyjścia jest zawsze najnowszą wersją deskryptora MQMD obsługiwaną przez menedżer kolejek, który wywołuje wyjście. Jeśli wyjście ma być przenośne między różnymi środowiskami, powinno ono zaznaczyć pole *MDVER* w pliku *MQMD* , aby sprawdzić, czy pola, do których musi uzyskać dostęp wyjście, znajdują się w strukturze.

W systemie IBM ido wyjścia jest przekazywany deskryptor MQMD version-2 .

Jeśli konwersja zakończyła się pomyślnie, w danych wyjściowych pola *MDENC* i *MDCSI* powinny zostać zmienione na wartości żądane przez aplikację. Zmiany te zostaną odzwierciedlone w aplikacji. Wszelkie inne zmiany wprowadzone przez wyjście do struktury są ignorowane i nie są odzwierciedlane w aplikacji.

Jeśli wyjście zwraca wartość XROK w polu *DXRES* struktury MQDXP, ale nie zmienia pól *MDENC* lub *MDCSI* w deskryptorze komunikatu, menedżer kolejek zwraca dla tych pól wartości, które odpowiednie pola w strukturze MQDXP miały na wejściu do wyjścia.

INLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość w bajtach *INBUF*.

Jest to długość buforu wejściowego *INBUF*i określa liczbę bajtów, które mają być przetworzone przez wyjście. *INLEN* jest mniejszą z długości danych komunikatu przed konwersją oraz długości buforu udostępnianego przez aplikację w wywołaniu MQGET.

Wartość jest zawsze większa od zera.

INBUF (1-bajtowy łańcuch bitowy x INLEN)-wejście

Bufor zawierający nieprzekształcony komunikat.

Zawiera dane komunikatu przed konwersją. Jeśli wyjście nie może przekształcić danych, menedżer kolejek zwraca treść tego buforu do aplikacji po zakończeniu wyjścia.

Uwaga: Wyjście nie powinno zmieniać *INBUF* ; Jeśli ten parametr zostanie zmieniony, wyniki są niezdefiniowane.

OUTLEN (10-cyfrowa liczba całkowita ze znakiem)-wejście

Długość w bajtach *OUTBUF*.

Jest to długość buforu wyjściowego *OUTBUF*i jest taka sama jak długość buforu udostępnianego przez aplikację w wywołaniu MQGET.

Wartość jest zawsze większa od zera.

OUTBUF (1-bajtowy łańcuch x OUTLEN)-wyjście

Bufor zawierający przekształcony komunikat.

W przypadku danych wyjściowych z wyjścia, jeśli konwersja zakończyła się pomyślnie (wskazywana przez wartość XROK w polu *DXRES* parametru **MQDXP**), **OUTBUF** zawiera dane komunikatu, które mają zostać dostarczone do aplikacji, w żądanej reprezentacji. Jeśli konwersja nie powiodła się, wszystkie zmiany wprowadzone przez wyjście w tym buforze są ignorowane.

Wywołanie RPG (ILE)

```
C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
C                     CALLP     exitname(MQDXP : MQMD : INLEN :
C                                        INBUF : OUTLEN : OUTBUF)
```

Definicja prototypu dla wywołania jest następująca:

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
Dexitname         PR                  EXTPROC('exitname')
D* Data-conversion exit parameter block
D MQDXP                          44A
D* Message descriptor
D MQMD                          364A
D* Length in bytes of INBUF
D INLEN                          10I 0 VALUE
D* Buffer containing the unconverted message
D INBUF                           *   VALUE
D* Length in bytes of OUTBUF
D OUTLEN                         10I 0 VALUE
D* Buffer containing the converted message
D OUTBUF                          *   VALUE
```

Koniec interfejsu programistycznego wrażliwego na działanie produktu

Informacje dodatkowe o wyjściach użytkownika, wyjściach funkcji API i instalowalnych usługach

Informacje znajdujące się w tej sekcji są pomocne podczas tworzenia programów zewnętrznych, programów zewnętrznych funkcji API i aplikacji usług, które można zainstalować:

- “Struktura MQIEP” na stronie 1502
- “Odwołanie do wyjścia konwersji danych” na stronie 1506
- “MQ_PUBLISH_EXIT-Wyjście publikowania” na stronie 1510
- “Wywołania wyjścia kanału i struktury danych” na stronie 1518
- “Odwołanie do wyjścia funkcji API” na stronie 1610
- “Informacje uzupełniające o interfejsie usług instalowalnych” na stronie 1671

Pojęcia pokrewne

Wyjścia użytkownika, wyjścia funkcji API i instalowalne usługi systemu IBM MQ

Zadania pokrewne

Rozszerzanie narzędzi menedżera kolejek

Struktura MQIEP

Struktura MQIEP zawiera punkt wejścia dla każdego wywołania funkcji, które może być wykonywane przez wyjścia.

Pola

StrucId

Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQIEP_STRUC_ID (identyfikator struktury MQ)

Wersja

Typ: MQLONG-input

Numer wersji struktury. Wartość jest następująca:

MQIEP_VERSION_1
Numer wersji struktury wersji 1.

MQIEP_CURRENT_VERSION
Bieżąca wersja struktury.

StrucLength
Typ: MQLONG

Wielkość struktury MQIEP w bajtach. Wartość jest następująca:

MQIEP_LENGTH_1

Flagi
Typ: MQLONG

Udostępnia informacje o adresach funkcji. Flaga wskazująca, czy biblioteka jest wielowątkowa, może być użyta z flagą wskazującą, czy biblioteka jest biblioteką klienta czy serwera.

Do określenia braku informacji o bibliotece używana jest następująca wartość:

MQIEPF_BRAK

Do określenia, czy biblioteka współużytkowana jest wielowątkowa, czy niewielowątkowa, używana jest jedna z następujących wartości:

MQIEPF_NON_THREADED_LIBRARY
Niewątkowa biblioteka współużytkowana

MQIEPF_THREADED_LIBRARY (BIBLIOTEKA MQIEPF_THREADED_
Wielowątkowa biblioteka współużytkowana

Do określenia, czy biblioteka współużytkowana jest biblioteką współużytkowaną klienta, czy serwera, używana jest jedna z następujących wartości:

MQIEPF_CLIENT_LIBRARY
Biblioteka współużytkowana klienta

Biblioteka MQIEPF_LOCAL_LIBRARY
Biblioteka współużytkowana serwera

Zarezerwowane
Typ: MQPTR

Wywołanie MQBACK_Call
Typ: PMQ_BACK_CALL

Adres wywołania MQBACK.

Wywołanie MQBEGIN_Call
Typ: PMQ_BEGIN_CALL

Adres wywołania komendy MQBEGIN.

Wywołanie MQBUFMH
Typ: PMQ_BUFMH_CALL

Adres wywołania MQBUFMH.

Wywołanie MQCB_Call
Typ: PMQ_CB_CALL

Adres wywołania MQCB.

Wywołania MQCLOSE_Call
Typ: PMQ_CLOSE_CALL

Adres wywołania MQCLOSE.

Wywołanie MQCMIT_Call
Typ: PMQ_CMIT_CALL

Adres wywołania MQCMIT.

Wywołanie MQCONN_Call
Typ: PMQ_CONN_CALL

Adres wywołania MQCONN.

Wywołanie MQCONNX_Call
Typ: PMQ_CONNX_CALL

Adres wywołania MQCONNX.

MQCRTMH_Wywołanie
Typ: PMQ_CRTMH_CALL

Adres wywołania MQCRTMH.

Wywołanie MQCTL_Call
Typ: PMQ_CTL_CALL

Adres wywołania MQCTL.

Wywołanie MQDISC
Typ: PMQ_DISC_CALL

Adres wywołania MQDISC.

Wywołanie MQDLTMH
Typ: PMQ_DLTMH_CALL

Adres wywołania MQDLTMH.

Wywołanie MQDLTMP_Call
Typ: PMQ_DLTMP_CALL

Adres wywołania MQDLTMP.

Wywołanie MQGET_Call
Typ: PMQ_GET_CALL

Adres wywołania MQGET.

Wywołania MQINQ_Call
Typ: PMQ_INQ_CALL

Adres wywołania MQINQ.

Wywołanie MQINQMP_Call
Typ: PMQ_INQMP_CALL

Adres wywołania MQINQMP.

MQMHBUF_Wywołanie
Typ: PMQ_MHBUF_CALL

Adres wywołania MQMHBUF.

Wywołanie MQOPEN_Call
Typ: PMQ_OPEN_CALL

Adres wywołania MQOPEN.

Wywołanie MQPUT_Call
Typ: PMQ_PUT_CALL

Adres wywołania MQPUT.

MQPUT1_Call
Typ: PMQ_PUT1_CALL

Adres wywołania MQPUT1 .

Wywołanie MQSET
Typ: PMQ_SET_CALL

Adres wywołania MQSET.

Wywołanie MQSETMP_Call
Typ: PMQ_SETMP_CALL

Adres wywołania MQSETMP.

Wywołanie MQSTAT_Call
Typ: PMQ_STAT_CALL

Adres wywołania MQSTAT.

Wywołanie MQSUB_Call
Typ: PMQ_SUB_CALL

Adres wywołania MQSUB.

Wywołanie MQSUBRQ
Typ: PMQ_SUBRQ_CALL

Adres wywołania MQSUBRQ.

Wywołanie MQXCNVC
Typ: PMQ_XCNVC_CALL

Adres wywołania MQXCNVC.

Wywołanie MQXCLWLN_Call
Typ: PMQ_XCLWLN_CALL

Adres wywołania MQXCLWLN.

Wywołanie MQXDX_Call
Typ: PMQ_XDX_CALL

Adres wywołania MQXDX.

Wywołanie MQXEP_Call
Typ: PMQ_XEP_CALL

Adres wywołania MQXEP.

Wywołanie MQZEP_Call
Typ: PMQ_ZEP_CALL

Adres wywołania MQZEP.

Deklaracja 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 */
};
```

Odwołanie do wyjścia konwersji danych

W przypadku systemu z/OSkonieczne jest napisanie wyjść konwersji danych w języku asemblera. W przypadku innych platform zaleca się używanie języka programowania C.

Aby ułatwić tworzenie programu obsługi wyjścia konwersji danych, dostarczane są następujące zasoby:

- Szkielet pliku źródłowego
- Wywołanie konwersji znaków
- Program narzędziowy tworzący fragment kodu, który wykonuje konwersję danych na strukturach typów danych. Ten program narzędziowy pobiera tylko dane wejściowe w języku C. W systemie z/OSgeneruje kod asemblera.

Procedurę pisania programów można znaleźć w następujących sekcjach:

- IBM i Zapisywanie programu obsługi wyjścia konwersji danych w systemie IBM MQ for IBM i
- z/OS Zapisywanie programu obsługi wyjścia konwersji danych w systemie IBM MQ for z/OS
- Zapisywanie wyjścia konwersji danych w systemach IBM MQ for AIX or Linux
- Zapisywanie wyjścia konwersji danych w systemie IBM MQ for Windows

Plik źródłowy szkieletu

Można ich użyć jako punktu początkowego podczas pisania programu obsługi wyjścia konwersji danych.

Dostarczone pliki są wymienione w sekcji Tabela 816 na stronie 1506.

Tabela 816. Szkieletowe pliki źródłowe

Platforma	**Plik**
AIX AIX	amqsvfc0.c
IBM i IBM i	QMQMSAMP/QCSRC (AMQSVFC4)
Linux Linux	amqsvfc0.c
Systemy Windows Windows	amqsvfc0.c
z/OS z/OS	CSQ4BAX8 (“1” na stronie 1507) CSQ4BAX9 (“2” na stronie 1507) CSQ4CAX9 (“3” na stronie 1507)

Tabela 816. Szkieletowe pliki źródłowe (kontynuacja)	
Platforma	**Plik**
Uwagi: 1. Ilustruje wywołanie MQXCVNC. 2. Opakowanie fragmentów kodu wygenerowanych przez program narzędziowy do użytku we wszystkich środowiskach z wyjątkiem środowiska CICS. 3. Opakowanie fragmentów kodu wygenerowanych przez program narzędziowy w celu użycia w środowisku CICS .	

Wywołanie konwersji znaków

Wywołanie MQXCNVC (konwersja znaków) z programu obsługi wyjścia konwersji danych służy do przekształcania danych komunikatu znakowego z jednego zestawu znaków na inny. W przypadku niektórych zestawów znaków wielobajtowych (na przykład zestawów znaków UTF-16) należy użyć odpowiednich opcji.

Z poziomu wyjścia nie można wykonać żadnych innych wywołań MQI. Próba wykonania takiego wywołania kończy się niepowodzeniem z kodem przyczyny MQRC_CALL_IN_PROGRESS.

Więcej informacji na temat wywołania MQXCNVC i odpowiednich opcji zawiera sekcja “MQXCNVC- przekształcanie znaków” na stronie 950 .

Program narzędziowy do tworzenia kodu wyjścia konwersji

Ten temat zawiera więcej informacji na temat tworzenia kodu wyjścia konwersji.

Komendy służące do tworzenia kodu wyjścia konwersji są następujące:

IBM i **IBM i**
CVTMQMDTA (Konwersja typu danych IBM MQ)

ALW **Systemy AIX, Linux, and Windows**
`crtmqcvx` (Utwórz IBM MQ wyjście konwersji)

z/OS **z/OS**
CSQUCVX

Komenda dla danej platformy tworzy fragment kodu, który wykonuje konwersję danych na strukturach typów danych, do użycia w programie obsługi wyjścia konwersji danych. Komenda pobiera plik zawierający co najmniej jedną definicję struktury języka C. z/OS W systemie z/OSgeneruje on następnie zestaw danych zawierający fragmenty kodu asemblera i funkcje konwersji. Na innych platformach generuje plik z funkcją w języku C w celu przekształcenia każdej definicji struktury. W systemie z/OSprogram narzędziowy wymaga dostępu do biblioteki środowiska wykonawczego LE/370 SCEERUN.

Wywoływanie programu narzędziowego CSQUCVX w systemie z/OS

z/OS

Rysunek 10 na stronie 1508 przedstawia przykład kodu JCL używanego do wywoływania programu narzędziowego CSQUCVX.

```
//CVX    EXEC PGM=CSQUCVX
//STEPLIB  DD DISP=SHR,DSN=thlqual.SCSQANLE
//         DD DISP=SHR,DSN=thlqual.SCSQLOAD
//         DD DISP=SHR,DSN=le370qual.SCEERUN
//SYSPRINT DD SYSOUT=*
//CSQUINP  DD DISP=SHR,DSN=MY.MQSERIES.FORMATS(MSG1)
//CSQUOUT  DD DISP=OLD,DSN=MY.MQSERIES.EXITS(MSG1)
```

Rysunek 10. Przykładowy kod JCL używany do wywoływania programu narzędziowego CSQUCVX

Instrukcje definicji danych z/OS

z/OS

Program narzędziowy CSQUCVX wymaga instrukcji DD o następujących nazwach DD, które przedstawia Tabela 817 na stronie 1508:

Tabela 817. Nazwy i opisy instrukcji definicji danych

Instrukcja DD	Opis
SYSPRINT	Określa zestaw danych lub klasę buforu wydruku dla raportów i komunikatów o błędach.
CSQUINP	Określa partycjonowany zestaw danych zawierający definicje struktur danych do przekształcenia.
CSQUOUT	Określa partycjonowany zestaw danych, w którym mają zostać zapisane fragmenty kodu konwersji. Długość rekordu logicznego (LRECL) musi wynosić 80, a format rekordu (RECFM) musi mieć wartość FB.

Komunikaty o błędach w systemach AIX, Linux, and Windows

Komenda `crtmqcvx` zwraca komunikaty z zakresu od AMQ7953 do AMQ7970.

Te komunikaty są wymienione w sekcji Komunikaty i kody przyczyny *Komunikaty IBM MQ*.

Istnieją dwa główne typy błędów:

- Poważne błędy, takie jak błędy składniowe, gdy przetwarzanie nie może być kontynuowane.

  Na ekranie zostanie wyświetlony komunikat z numerem wiersza błędu w pliku wejściowym. Plik wyjściowy mógł zostać częściowo utworzony.
- Inne błędy, gdy wyświetlany jest komunikat informujący, że znaleziono problem, ale analizowanie struktury może być kontynuowane.

  Plik wyjściowy został utworzony i zawiera informacje o błędzie dotyczące problemów, które wystąpiły. Ta informacja o błędzie jest poprzedzona przedrostkiem `#error` , dzięki czemu wygenerowany kod nie jest akceptowany przez żaden kompilator bez interwencji w celu rozwiązania problemów.

Poprawna składnia

Plik wejściowy programu narzędziowego musi być zgodny ze składnią języka C.

Jeśli użytkownik nie zna języka C, należy zapoznać się z sekcją Przykład w języku C w tym temacie.

Ponadto należy pamiętać o następujących zasadach:

- Słowo kluczowe typedef jest rozpoznawane tylko przed słowem kluczowym struct.
- W deklaracjach struktury wymagany jest znacznik struktury.

- Można użyć pustego nawiasu kwadratowego [], aby oznaczyć tablicę lub łańcuch o zmiennej długości na końcu komunikatu.
- Tablice wielowymiarowe i tablice łańcuchów nie są obsługiwane.
- Rozpoznawane są następujące dodatkowe typy danych:
  - MQBOOL
  - MQBYTE
  - MQCHAR
  - MQFLOAT32
  - MQFLOAT64
  - Zmaterializowana tabela zapytania
  - MQLONG
  - MQINT8
  - MQUINT8
  - MQINT16
  - MQUINT16
  - MQINT32
  - MQUINT32
  - MQINT64
  - MQUINT64

  Pola MQCHAR są przekształcane w stronę kodową, ale pola MQBYTE, MQINT8 i MQUINT8 pozostają niezmienione. Jeśli kodowanie jest inne, odpowiednio przekształcane są: MQSHORT, MQLONG, MQINT16, MQUINT16, MQINT32, MQUINT32, MQINT64, MQUINT64, MQFLOAT32, MQFLOAT64 i MQBOOL.
- Nie należy używać następujących typów danych:
  - double (podwójna)
  - wskaźniki
  - pola bitowe

  Jest to spowodowane tym, że program narzędziowy do tworzenia kodu wyjścia konwersji nie udostępnia narzędzia do przekształcania tych typów danych. Aby temu zaradzić, można napisać własne procedury i wywołać je z wyjścia.

Inne punkty do odnotować:

- Nie używaj numerów kolejnych w zestawie danych wejściowych.
- Jeśli istnieją pola, dla których mają być dostępne własne procedury konwersji, należy je zadeklarować jako MQBYTE, a następnie zastąpić wygenerowane makra CMQXCFBA własnym kodem konwersji.

Przykład C

```
struct TEST { MQLONG    SERIAL_NUMBER;
              MQCHAR    ID[5];
              MQINT16   VERSION;
              MQBYTE    CODE[4];
              MQLONG    DIMENSIONS[3];
              MQCHAR    NAME[24];
            } ;
```

Odpowiada to następującym deklaracjom w innych językach programowania:

kompilatory

```
      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

Obsługiwane tylko w systemie 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-Wyjście publikowania

Wywołanie MQ_PUBLISH_EXIT może sprawdzać i zmieniać komunikaty dostarczane do subskrybentów.

Przeznaczenie

Użyj wyjścia publikowania, aby sprawdzić i zmienić komunikaty dostarczane do subskrybentów:

- Sprawdzanie treści komunikatu opublikowanego dla każdego subskrybenta
- Modyfikowanie treści komunikatu opublikowanego dla każdego subskrybenta
- Zmiana kolejki, w której umieszczany jest komunikat
- Zatrzymaj dostarczanie komunikatu do subskrybenta

To wyjście nie jest dostępne w systemie IBM MQ for z/OS.

Składnia

MQ_PUBLISH_EXIT (*ExitParms*, *PubContext*, *SubContext*)

Parametry

***ExitParms* (MQPSXP)- Input/Output**
ExitParms zawiera informacje o wywołaniu wyjścia.

***PubContext* (MQPBC) - Input**
PubContext zawiera informacje kontekstowe na temat publikatora publikacji.

***SubContext* (MQSBC) - Input/Output**
SubContext zawiera informacje kontekstowe o subskrybencie odbierającym publikację.

MQPSXP-struktura danych wyjścia publikowania

Struktura MQPSXP opisuje informacje, które są przekazywane i zwracane z wyjścia publikowania.

Tabela 818 na stronie 1511 zawiera podsumowanie pól w strukturze:

Tabela 818. Pola w MQPSXP

Pole	Opis
StrucID	Identyfikator struktury
Version	Numer wersji struktury
ExitId	Typ wywoływanego wyjścia
ExitReason	Przyczyna wywołania wyjścia
ExitResponse	Odpowiedź z wyjścia
ExitResponse2	Dodatkowa odpowiedź z wyjścia
Feedback	Kod zwrotny
ExitUserArea	Wyjdź z obszaru użytkownika
ExitData	Dane wyjścia
QMgrName	Nazwa lokalnego menedżera kolejek
Hconn	Uchwyt połączenia
MsgDescPtr	Adres deskryptora komunikatu (MQMD)
MsgHandle	Uchwyt do właściwości komunikatu (MQHMSG)
MsgInPtr	Adres komunikatu wejściowego
MsgInLength	Długość komunikatu wejściowego
MsgOutPtr	Adres komunikatu wyjściowego
MsgOutLength	Długość komunikatu wyjściowego
pEntryPoints	Adres struktury MQIEP

Pola

***StrucID* (MQCHAR4)**
StrucID jest identyfikatorem struktury. Wartość jest następująca:

MQPSXP_STRUCID
MQPSXP_STRUCID jest identyfikatorem struktury parametru wyjścia publikowania. Dla języka programowania C zdefiniowana jest również stała MQPSXP_STRUC_ID_ARRAY , która ma taką samą wartość jak MQPSXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucID to pole wejściowe do wyjścia.

***Version* (MQLONG)**
Version jest numerem wersji struktury. Wartość jest następująca:

MQPSXP_VERSION_1
MQPSXP_VERSION_1 to struktura parametru wyjścia publikowania w wersji 1. Stała MQPSXP_CURRENT_VERSION jest również zdefiniowana z tą samą wartością.

Version to pole wejściowe do wyjścia.

***ExitId* (MQLONG)**
ExitId jest typem wywoływanego wyjścia. Wartość jest następująca:

MQXT_PUBLISH_EXIT
Wyjście publikowania.

ExitId to pole wejściowe do wyjścia.

***ExitReason* (MQLONG)**
ExitReason jest przyczyną wywołania wyjścia. Możliwe wartości:

MQXR_INIT
Wyjście dla tego połączenia jest wywoływane w celu zainicjowania. Wyjście może uzyskać i zainicjować potrzebne zasoby, na przykład pamięć główną.

MQXR_TERM
Wyjście dla tego połączenia jest wywoływane, ponieważ wyjście ma zostać zatrzymane. Wyjście musi zwolnić wszystkie zasoby, które uzyskało od czasu zainicjowania, na przykład pamięć główną.

MQXR_PUBLICATION
Wyjście jest wywoływane przez menedżer kolejek przed umieszczeniem publikacji w kolejce komunikatów subskrybenta. Wyjście może zmienić komunikat, a nie umieścić go w kolejce lub zatrzymać publikację.

ExitReason to pole wejściowe do wyjścia.

***ExitResponse* (MQLONG)**
Ustaw wartość *ExitResponse* w wyjściu, aby określić, w jaki sposób przetwarzanie musi być kontynuowane. *ExitResponse* może przyjmować jedną z następujących wartości:

MQXCC_OK
Ustaw parametr MQXCC_OK , aby kontynuować przetwarzanie normalnie. Ustaw parametr MQXCC_OK w odpowiedzi na dowolną wartość parametru ExitReason.

Jeśli parametr ExitReason ma wartość MQXR_PUBLICATION, pola *DestinationQName* i *DestinationQMgrName* w strukturze MQSBC identyfikują miejsce docelowe, do którego jest wysyłany komunikat.

MQXCC_FAILED
Ustaw wartość MQXCC_FAILED , aby zatrzymać operację publikowania. Kod zakończenia MQCC_FAILED i kod przyczyny 2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR jest ustawiany po powrocie z wyjścia.

MQXCC_SUPPRESS_FUNCTION
Ustaw wartość MQXCC_SUPPRESS_FUNCTION , aby zatrzymać normalne przetwarzanie komunikatu. Parametr MQXCC_SUPPRESS_FUNCTION należy ustawić tylko wtedy, gdy parametr *ExitReason* ma wartość MQXR_PUBLICATION.

Komunikat jest nadal przetwarzany przez menedżer kolejek zgodnie z opcją MQRO_DISCARD_MSG w polu *Report* w deskryptorze komunikatu.

- Jeśli zostanie podana opcja MQRO_DISCARD_MSG , komunikat nie zostanie dostarczony do subskrybenta.
- Jeśli opcja MQRO_DISCARD_MSG nie jest określona, komunikat jest umieszczany w kolejce niedostarczonych komunikatów. Jeśli nie ma kolejki niedostarczonych komunikatów lub komunikat nie może zostać pomyślnie umieszczony w kolejce niedostarczonych komunikatów, publikacja nie zostanie dostarczona do subskrybenta. Dostarczenie publikacji do innych subskrybentów zależy od wartości atrybutów obiektu tematu PMSGDLV i NPMSGDLV . Opis tych atrybutów zawiera opis parametrów komendy DEFINE TOPIC .

ExitResponse jest polem wyjściowym z wyjścia.

***ExitResponse2* (MQLONG)**
Produkt *ExitResponse2* jest zarezerwowany do użycia w przyszłości.

***Feedback* (MQLONG)**
Feedback to kod informacji zwrotnej, który ma być używany, gdy wyjście zwraca wartość MQXCC_SUPPRESS_FUNCTION w *ExitResponse*.

Na wejściu do wyjścia *Feedback* zawsze ma wartość MQFB_NONE. Jeśli wyjście zwraca wartość MQXCC_SUPPRESS_FUNCTION, należy ustawić parametr *Feedback* na wartość, która ma być używana dla komunikatu, gdy menedżer kolejek umieści go w kolejce niedostarczonych komunikatów. W przypadku powrotu z wyjścia, jeśli parametr *Feedback* ma pierwotną wartość MQFB_NONE, menedżer kolejek ustawia parametr *Feedback* na wartość MQFB_STOPPED_BY_PUBSUB_EXIT.

Feedback to pole wejściowe/wyjściowe do wyjścia.

***ExitUserArea* (MQBYTE16)**
ExitUserArea to pole, które jest dostępne do użycia przez wyjście. Każde połączenie ma osobną *ExitUserArea*. Długość parametru *ExitUserArea* jest określona przez wartość MQ_EXIT_USER_AREA_LENGTH.

Pole *ExitReason* ma wartość MQXR_INIT przy pierwszym wywołaniu wyjścia. Parametr *ExitUserArea* jest inicjowany jako MQXUA_NONE przy pierwszym wywołaniu wyjścia dla połączenia. Kolejne zmiany w pliku *ExitUserArea* są zachowywane między wywołaniami wyjścia.

ExitUserArea to pole wejściowe/wyjściowe do wyjścia.

***ExitData* (MQCHAR32)**
ExitData to stałe dane wyjścia zdefiniowane przez parametr **PublishExitData** w sekcji pliku inicjowania menedżera kolejek. Dane są dopełniane spacjami do pełnej długości pola. Jeśli w pliku inicjowania nie ma zdefiniowanych stałych danych wyjścia, pole *ExitData* jest puste. Długość parametru *ExitData* jest określona przez wartość MQ_EXIT_DATA_LENGTH.

ExitData to pole wejściowe do wyjścia.

***QMgrName* (MQCHAR48)**
QMgrName jest nazwą lokalnego menedżera kolejek. Nazwa jest dopełniana spacjami do pełnej długości pola. Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

QMgrName to pole wejściowe do wyjścia.

***Hconn* (MQHCONN)**
Hconn jest uchwytem reprezentującym połączenie z menedżerem kolejek. Aby pracować z właściwościami komunikatu, należy używać parametru *Hconn* tylko w wywołaniach funkcji właściwości komunikatu MQSETMP, MQINQMMPlub MQDLTMP .

Hconn to pole wejściowe do wyjścia.

***MsgDescPtr* (PMQMD)**
MsgDescPtr jest adresem deskryptora komunikatu (MQMD) przetwarzanego komunikatu i jest kopią deskryptora MQMD zwróconego z wywołania MQPUT. Wyjście może zmienić treść deskryptora komunikatu. Wszelkie zmiany w treści deskryptora komunikatu muszą być wykonywane z ostrożnością. W szczególności w przypadku, gdy pole *SubType* struktury MQSBC ma wartość MQSUBTYPE_PROXY, pole *CorrelId* w deskryptorze komunikatu nie może być zmieniane.

Do wyjścia nie jest przekazywany żaden deskryptor komunikatu, jeśli *ExitReason* ma wartość MQXR_INIT lub MQXR_TERM . W takich przypadkach *MsgDescPtr* jest wskaźnikiem pustym.

MsgDescPtr to pole wejściowe do wyjścia.

***MsgHandle* (MQHMSG)**
MsgHandle jest uchwytem właściwości komunikatu. Funkcji *MsgHandle* należy używać tylko z wywołaniami funkcji właściwości komunikatu MQSETMP, MQINQMMPlub MQDLTMP , aby pracować z właściwościami komunikatu.

MsgHandle to pole wejściowe do wyjścia.

***MsgInPtr* (PMQVOID)**
MsgInPtr jest adresem danych komunikatu wejściowego. Zawartość buforu adresowanego przez *MsgInPtr* może być modyfikowana przez wyjście; patrz *MsgOutPtr* .

MsgInPtr to pole wejściowe do wyjścia.

***MsgInLength* (MQLONG)**
MsgInLength jest długością (w bajtach) danych komunikatu przekazanych do wyjścia. Adres danych jest podawany przez *MsgInPtr*.

MsgInLength to pole wejściowe do wyjścia.

***MsgOutPtr* (PMQVOID)**
MsgOutPtr jest adresem buforu zawierającego dane komunikatu zwracane przez wyjście. Przy wejściu do wyjścia *MsgOutPtr* ma wartość null. W przypadku powrotu z wyjścia, jeśli wartość nadal wynosi NULL, menedżer kolejek wysyła komunikat określony przez parametr *MsgInPtr*o długości określonej przez parametr *MsgInLength*.

Jeśli wyjście modyfikuje dane komunikatu, użyj jednej z następujących procedur:

- Jeśli długość danych nie zmienia się, dane mogą być modyfikowane w buforze adresowanym przez *MsgInPtr*. W takim przypadku nie należy zmieniać wartości *MsgOutPtr* i *MsgOutLength*.
- Jeśli zmodyfikowane dane są krótsze niż dane oryginalne, można je zmodyfikować w buforze adresowanym przez program *MsgInPtr*. W tym przypadku parametr *MsgOutPtr* musi być ustawiony na adres buforu komunikatu wejściowego, a parametr *MsgOutLength* na nową długość danych komunikatu.
- Jeśli zmodyfikowane dane są lub mogą być dłuższe niż oryginalne dane, wyjście musi uzyskać nowy bufor komunikatów. Skopiuj do niego zmodyfikowane dane. Ustaw parametr *MsgOutPtr* na adres nowego buforu, a parametr *MsgOutLength* na długość danych nowego komunikatu. Wyjście jest odpowiedzialne za zwolnienie buforu adresowanego przez program *MsgOutPtr* podczas następnego wywołania wyjścia.

Uwaga: *MsgOutPtr* jest zawsze wskaźnikiem pustym na wejściu do wyjścia, a nie adresem poprzednio uzyskanego buforu komunikatów. Aby zwolnić poprzednio uzyskany bufor, wyjście musi zapisać swój adres i długość. Zapisz informacje w pliku *ExitUserArea*lub w bloku kontrolnym, którego adres jest zapisany w pliku *ExitUserArea*.

MsgOutPtr to pole wejściowe/wyjściowe do wyjścia.

***MsgOutLength* (MQLONG)**
MsgOutLength to długość (w bajtach) danych komunikatu zwróconych przez wyjście. Na wejściu do wyjścia to pole ma zawsze wartość zero. Po powrocie z wyjścia to pole jest ignorowane, jeśli parametr *MsgOutPtr* ma wartość NULL. Informacje na temat modyfikowania danych komunikatu zawiera sekcja *MsgOutPtr* .

MsgOutLength to pole wejściowe/wyjściowe do wyjścia.

***pEntryPoints* (PMQIEP)**
pEntryPoints to adres struktury MQIEP, za pośrednictwem której można wykonywać wywołania MQI i DCI.

Deklaracja języka 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-struktura danych kontekstu publikacji

Struktura MQPBC zawiera informacje kontekstowe dotyczące publikatora publikacji, które są przekazywane do wyjścia publikowania.

Tabela 819 na stronie 1515 zawiera podsumowanie pól w strukturze:

Tabela 819. Pola w MQPBC	
Pole	**Opis**
StrucID	Identyfikator struktury
Version	Numer wersji struktury
PubTopicString	Publikuj łańcuch tematu
MsgDescPtr	Adres deskryptora komunikatu (MQMD)

Pola

***StrucID* (MQCHAR4)**
StrucID jest identyfikatorem struktury. Wartość jest następująca:

MQPBC_STRUCID
MQPBC_STRUCID to identyfikator struktury kontekstu publikacji. Dla języka programowania C zdefiniowana jest również stała MQPBC_STRUC_ID_ARRAY , która ma taką samą wartość jak MQPBC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucID to pole wejściowe do wyjścia.

***Version* (MQLONG)**
Version jest numerem wersji struktury. Wartość jest następująca:

MQPBC_VERSION_1
MQPBC_VERSION_1 to struktura parametru wyjścia publikowania w wersji 1.

MQPBC_VERSION_2
MQPBC_VERSION_2 to struktura parametru wyjścia publikowania w wersji 2. Stała MQPBC_CURRENT_VERSION jest również zdefiniowana z tą samą wartością.

Version to pole wejściowe do wyjścia.

***PubTopicString* (MQCHARV)**
PubTopicString to łańcuch tematu, w którym jest publikowany.

PubTopicString to pole wejściowe do wyjścia.

***MsgDescPtr* (PMQMD)**
MsgDescPtr jest adresem kopii deskryptora komunikatu (MQMD) dla przetwarzanego komunikatu.

MsgDescPtr to pole wejściowe do wyjścia.

Deklaracja języka 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-struktura danych kontekstu subskrypcji

Struktura MQSBC zawiera informacje kontekstowe związane z subskrybentem odbierającym publikację, które są przekazywane do wyjścia publikowania.

Tabela 820 na stronie 1516 zawiera podsumowanie pól w strukturze:

Tabela 820. Zmienne w MQSBC	
Pole	**Opis**
StrucID	Identyfikator struktury
Version	Numer wersji struktury
DestinationQMgrName	Nazwa docelowego menedżera kolejek
DestinationQName	Nazwa kolejki docelowej
SubType	Typ subskrypcji.
SubOptions	Opcje subskrypcji
ObjectName	Nazwa obiektu
ObjectString	Łańcuch obiektu
SubTopicString	Łańcuch tematu subskrypcji
SubName	Nazwa subskrypcji
SubId	Identyfikator subskrypcji
SelectionString	Adres łańcucha wyboru
SubLevel	Poziom subskrypcji
PSProperties	Właściwości publikowania/subskrybowania

Pola

***StrucID* (MQCHAR4)**
Identyfikator struktury. Wartość jest następująca:

MQSBC_STRUCID
MQSBC_STRUCID jest identyfikatorem struktury parametru wyjścia publikowania.
W języku programowania C stała MQSBC_STRUC_ID_ARRAY jest również zdefiniowana; MQSBC_STRUC_ID_ARRAY ma taką samą wartość jak MQSBC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

StrucID to pole wejściowe do wyjścia.

***Version* (MQLONG)**
Numer wersji struktury. Wartość jest następująca:

MQSBC_VERSION_1
Struktura parametru wyjścia publikowania w wersji 1. Stała MQSBC_CURRENT_VERSION jest również zdefiniowana z tą samą wartością.

Version to pole wejściowe do wyjścia.

***DestinationQMgrName* (MQCHAR48)**
DestinationQMgrName to nazwa menedżera kolejek, do którego wysyłany jest komunikat. Nazwa jest dopełniana spacjami do pełnej długości pola. Nazwa może być zmieniona przez wyjście. Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

DestinationQMgrName jest polem wejścia/wyjścia do wyjścia; patrz uwaga.

***DestinationQName* (MQCHAR48)**
DestinationQName jest nazwą kolejki, do której wysyłany jest komunikat. Nazwa jest dopełniana spacjami do pełnej długości pola. Nazwa może być zmieniona przez wyjście. Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH.

DestinationQName jest polem wejścia/wyjścia do wyjścia; patrz uwaga.

***SubType* (MQLONG)**
SubType wskazuje sposób utworzenia subskrypcji. Poprawne wartości to MQSUBTYPE_API, MQSUBTYPE_ADMIN i MQSUBTYPE_PROXY ; patrz Inquire Subscription Status (Response).

SubType to pole wejściowe do wyjścia.

***SubOptions* (MQLONG)**
SubOptions to opcje subskrypcji. Opis wartości, które może przyjmować to pole, zawiera sekcja “Opcje (MQLONG) dla MQSD” na stronie 589 .

SubOptions to pole wejściowe do wyjścia.

***ObjectName* (MQCHAR48)**
ObjectName jest nazwą obiektu tematu zdefiniowaną w lokalnym menedżerze kolejek. Długość tego pola jest określona przez wartość MQ_TOPIC_NAME_LENGTH. Nazwa obiektu jest nazwą obiektu tematu administracyjnego, który menedżer kolejek przypisał do łańcucha tematu. Nawet jeśli subskrybent udostępnili obiekt tematu jako część subskrypcji, parametr *ObjectName* może być innym obiektem tematu. Powiązanie obiektu tematu z subskrypcją zależy od pełnego rozstrzygnięcia parametru *SubTopicString*.

ObjectName to pole wejściowe do wyjścia.

***ObjectString* (MQCHARV)**
ObjectString to pełny łańcuch tematu publikacji, która została zasubskrybowana. Wszystkie znaki wieloznaczne w oryginalnym łańcuchu subskrypcji są rozstrzygane. Różni się ono od pola MQSD subscription *ObjectString* (subskrypcja) opisanego w sekcji “ObjectString (MQCHARV) dla MQSD” na stronie 600, które może zawierać znaki wieloznaczne i nie zawiera żadnej nazwy obiektu udostępnionej przez subskrybenta.

ObjectString to pole wejściowe do wyjścia.

***SubTopicString* (MQCHARV)**
SubTopicString to pełny łańcuch tematu dostarczony przez subskrybent. *SubTopicString* jest kombinacją łańcucha tematu zdefiniowanego w obiekcie tematu i łańcucha tematu. Subskrybent musi udostępnić obiekt tematu, łańcuch tematu lub oba te elementy. Jeśli subskrybent udostępnia łańcuch tematu, może zawierać znaki wieloznaczne.

SubTopicString to pole wejściowe do wyjścia.

***SubName* (MQCHARV)**
SubName jest nazwą subskrypcji, która jest udostępniana przez subskrybent, lub jest nazwą wygenerowaną.

SubName to pole wejściowe do wyjścia.

***SubId* (MQBYTE 24)**
SubId jest unikalnym wewnętrznym identyfikatorem subskrypcji.

SubId to pole wejściowe do wyjścia.

***SelectionString* (MQCHARV)**
SelectionString to kryteria wyboru używane podczas subskrybowania komunikatów z tematu. Więcej informacji na ten temat zawiera sekcja Selektory.

SelectionString to pole wejściowe do wyjścia.

***SubLevel* (MQLONG)**
SubLevel to poziom przechwytywania powiązany z subskrypcją. Więcej informacji na ten temat zawiera sekcja “SubLevel (MQLONG) dla MQSD” na stronie 603 .

SubLevel to pole wejściowe do wyjścia.

***PSProperties* (MQLONG)**
PSProperties to właściwości publikowania/subskrybowania. Określają one sposób dodawania właściwości komunikatu związanych z publikowaniem/subskrybowaniem do komunikatów wysyłanych do tej subskrypcji. Możliwe wartości to MQPSPROP_NONE, MQPSPROP_COMPAT, MQPSPROP_RFH2, MQPSPROP_MSGPROP. Opis tych wartości zawiera sekcja Parametry opcjonalne (zmiana, kopiowanie i tworzenie subskrypcji) .

PSProperties to pole wejściowe do wyjścia.

Uwaga: Sprawdzenia autoryzacji są wykonywane tylko na oryginalnych wartościach *DestinationQMgrName* i *DestinationQName* przed przekazaniem ich do wyjścia publikowania. Nowe sprawdzenia autoryzacji nie są wykonywane, gdy wyjście zmienia kolejkę docelową, zmieniając wartość *DestinationQMgrName* lub *DestinationQName*.

Deklaracja języka 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;
```

Wywołania wyjścia kanału i struktury danych

Ta kolekcja tematów zawiera informacje uzupełniające na temat specjalnych wywołań IBM MQ i struktur danych, których można używać podczas pisania programów obsługi wyjścia kanału.

Te informacje są informacjami interfejsu programistycznego wrażliwymi na działanie produktu. Procedury zewnętrzne IBM MQ można pisać w następujących językach programowania:

Tabela 821. Procedury zewnętrzne IBM MQ : platformy i języki programowania

Platforma	**Języki programowania**
IBM MQ for z/OS	Asembler i C (które muszą być zgodne ze środowiskiem programowania systemowego C dla wyjść systemowych, opisane w podręczniku *z/OS C/C++ Programming Guide*).
IBM MQ for IBM i	ILE C, ILE COBOL i ILE RPG
Wszystkie pozostałe platformy IBM MQ	C

Można również pisać procedury zewnętrzne w programie Java wyłącznie do użytku z aplikacjami Java i JMS . Więcej informacji na temat tworzenia i używania wyjść kanałów w produkcie IBM MQ classes for Javazawiera sekcja Korzystanie z wyjść kanałów w programie IBM MQ classes for Java , a w przypadku produktu IBM MQ classes for JMS-sekcja Korzystanie z wyjść kanałów w produkcie IBM MQ classes for JMS.

Nie można pisać programów zewnętrznych IBM MQ w języku TAL ani Visual Basic. Jednak deklaracja struktury MQCD jest dostępna w języku Visual Basic i może być używana w wywołaniu MQCONNX z programu IBM MQ MQI client .

W wielu przypadkach w kolejnych opisach parametry są tablicami lub łańcuchami znaków o nieustalonej wielkości. W przypadku tych parametrów do reprezentowania stałej liczbowej używana jest mała litera "n" . Jeśli deklaracja dla tego parametru jest zakodowana, wartość "n" musi zostać zastąpiona wymaganą wartością liczbową. Więcej informacji na temat konwencji używanych w tych opisach zawiera publikacja “Podstawowe typy danych” na stronie 236.

Pliki definicji danych

Pliki definicji danych są dostarczane z produktem IBM MQ dla każdego obsługiwanego języka programowania. Szczegółowe informacje na temat tych plików zawiera sekcja Kopiowanie, nagłówek, włączanie i pliki modułu.

MQ_CHANNEL_EXIT-wyjście kanału

Wywołanie MQ_CHANNEL_EXIT opisuje parametry, które są przekazywane do każdego wyjścia kanału wywoływanego przez agenta kanału komunikatów.

Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQ_CHANNEL_EXIT. Nazwa MQ_CHANNEL_EXIT nie ma specjalnego znaczenia, ponieważ nazwy wyjść kanału są udostępniane w definicji kanału MQCD.

Istnieje pięć typów wyjść kanału:

- Wyjście zabezpieczeń kanału
- Wyjście komunikatu kanału
- Wyjście wysyłania kanału
- Wyjście odbierania kanału
- Komunikat kanału-wyjście ponowienia

Parametry są podobne dla każdego typu wyjścia, a opis podany w tym miejscu ma zastosowanie do wszystkich z nich, z wyjątkiem przypadków, w których zaznaczono inaczej.

Składnia

MQ_CHANNEL_EXIT *(ChannelExitParms, ChannelDefinition, DataLength, AgentBufferLength, AgentBuffer, ExitBufferLength, ExitBufferAddr)*

Parametry

Wywołanie MQ_CHANNEL_EXIT ma następujące parametry.

ChannelExit-parametry wejściowe/wyjściowe (MQCXP)

Blok parametru wyjścia kanału.

Ta struktura zawiera dodatkowe informacje dotyczące wywołania wyjścia. Wyjście ustawia informacje w tej strukturze w celu wskazania sposobu działania agenta MCA.

ChannelDefinition (MQCD)-wejście/wyjście

Definicja kanału.

Ta struktura zawiera parametry ustawione przez administratora w celu sterowania zachowaniem kanału.

DataLength (MQLONG)-wejście/wyjście

Długość danych.

Dane zależą od typu wyjścia:

- W przypadku wyjścia zabezpieczeń kanału po wywołaniu wyjścia ten parametr zawiera długość dowolnego komunikatu zabezpieczeń w polu *AgentBuffer* , jeśli parametr *ExitReason* ma wartość MQXR_SEC_MSG. Jeśli nie ma komunikatu, ma wartość zero. Wyjście musi

ustawić w tym polu długość dowolnego komunikatu bezpieczeństwa, który ma zostać wysłany do partnera, jeśli parametr *ExitResponse* ma wartość MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG. Dane komunikatu znajdują się w katalogu *AgentBuffer* lub *ExitBufferAddr*.

Treść komunikatów bezpieczeństwa jest wyłączną odpowiedzialnością za wyjścia zabezpieczeń.

- W przypadku wyjścia komunikatów kanału po wywołaniu wyjścia ten parametr zawiera długość komunikatu (w tym nagłówek kolejki transmisji). Wyjście musi ustawić w tym polu długość komunikatu w *AgentBuffer* lub *ExitBufferAddr* , który ma być kontynuowany. Wartość ta musi być większa lub równa długości nagłówka kolejki transmisji (MQXQH).
- W przypadku wyjścia wysyłania lub odbierania kanału, gdy wyjście jest wywoływane, ten parametr zawiera długość transmisji. Wyjście musi ustawić w tym polu długość transmisji w *AgentBuffer* lub *ExitBufferAddr* , która ma być kontynuowana.

Jeśli wyjście zabezpieczeń wysyła komunikat, a na drugim końcu kanału nie ma wyjścia zabezpieczeń lub drugi koniec ustawia *ExitResponse* na wartość MQXCC_OK, wyjście inicjujące jest ponownie wywoływane z MQXR_SEC_MSG i odpowiedzią o wartości NULL (*DataLength* = 0).

AgentBufferLength (MQLONG)-wejście

Długość buforu agenta.

Ten parametr może być większy niż *DataLength* w wywołaniu.

W przypadku komunikatów kanału, wyjść wysyłania i odbierania wszystkie nieużywane miejsca w wywołaniu mogą być używane przez wyjście do rozwijania danych w miejscu. W takim przypadku parametr **DataLength** musi zostać odpowiednio ustawiony przez wyjście.

W języku programowania C parametr ten jest przekazywany przez adres.

AgentBuffer (MQBYTE x AgentBufferLength)-wejścia/wyjścia

Bufor agenta.

Zawartość tego parametru zależy od typu wyjścia:

- W przypadku wyjścia zabezpieczeń kanału po wywołaniu wyjścia zawiera on komunikat zabezpieczeń, jeśli parametr *ExitReason* ma wartość MQXR_SEC_MSG. Aby wysłać komunikat bezpieczeństwa z powrotem, wyjście może użyć tego buforu lub własnego buforu (*ExitBufferAddr*).
- W przypadku wyjścia komunikatu kanału, po wywołaniu wyjścia ten parametr zawiera:
  - Nagłówek kolejki transmisji (MQXQH), który zawiera deskryptor komunikatu (który sam zawiera informacje o kontekście komunikatu), po którym następuje bezpośrednio
  - Dane komunikatu

  Jeśli komunikat ma być kontynuowany, wyjście może wykonać jedną z następujących czynności:

  - Pozostaw zawartość buforu bez zmian
  - Zmodyfikuj istniejącą treść (zwracając nową długość danych w pliku *DataLength*). nie może być większa niż *AgentBufferLength*)
  - Skopiuj treść do pliku *ExitBufferAddr*, wprowadzając wymagane zmiany.

  Wszelkie zmiany wprowadzane przez wyjście w nagłówku kolejki transmisji nie są sprawdzane, jednak błędne modyfikacje mogą oznaczać, że komunikat nie może zostać umieszczony w miejscu docelowym.
- W przypadku wyjścia wysyłania lub odbierania kanału, po wywołaniu wyjścia zawiera ono dane transmisji. Wyjście może wykonać jedną z następujących czynności:
  - Pozostaw zawartość buforu bez zmian
  - Zmodyfikuj istniejącą treść (zwracając nową długość danych w pliku *DataLength*). nie może być większa niż *AgentBufferLength*)
  - Skopiuj treść do pliku *ExitBufferAddr*, wprowadzając wymagane zmiany.

Pierwsze 8 bajtów danych nie może być zmieniane przez wyjście.

ExitBufferLength (MQLONG)-wejścia/wyjścia

Długość buforu wyjścia.

Przy pierwszym wywołaniu wyjścia ten parametr jest ustawiany na zero. Następnie każda wartość przekazywana z powrotem przez wyjście, przy każdym wywołaniu, jest prezentowana do wyjścia przy następnym wywołaniu. Wartość nie jest używana przez agent MCA.

Uwaga: Ten parametr nie może być używany przez wyjścia napisane w językach programowania, które nie obsługują typu danych wskaźnika.

ExitBufferAddr (MQPTR)-wejście/wyjście

Adres buforu wyjścia.

Ten parametr jest wskaźnikiem do adresu buforu pamięci masowej zarządzanej przez wyjście, gdzie można wybrać zwracanie komunikatu lub danych transmisji (w zależności od typu wyjścia) do agenta, jeśli bufor agenta jest lub może nie być wystarczająco duży lub jeśli jest to wygodniejsze dla wyjścia.

Przy pierwszym wywołaniu wyjścia adres przekazany do wyjścia ma wartość NULL. Następnie adres przekazywany z powrotem przez wyjście, przy każdym wywołaniu, jest prezentowany do wyjścia przy następnym wywołaniu.

Jeśli parametr ExitBufferAddr ma wartość null, używane dane są pobierane z parametru AgentBuffer .

Jeśli parametr ExitBufferAddr nie ma wartości NULL, używane dane są pobierane z buforu wskazywanego przez parametr ExitBufferAddr.

Uwaga: Ten parametr nie może być używany przez wyjścia napisane w językach programowania, które nie obsługują typu danych wskaźnika.

Wywołanie C

```
exitname (&ChannelExitParms, &ChannelDefinition,
&DataLength, &AgentBufferLength, AgentBuffer,
&ExitBufferLength, &ExitBufferAddr);
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
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 */
```

Wywołanie COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION,
                           DATALENGTH, AGENTBUFFERLENGTH, AGENTBUFFER,
                           EXITBUFFERLENGTH, EXITBUFFERADDR.
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
**   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.
```

Wywołanie RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD : DATLEN :
     C                                       ABUFL : ABUF : EBUFL :
     C                                       EBUF)
```

Definicja prototypu dla wywołania jest następująca:

```
     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                             *
```

Wywołanie asemblera systemu System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION,DATALENGTH, X
               AGENTBUFFERLENGTH,AGENTBUFFER,EXITBUFFERLENGTH,         X
               EXITBUFFERADDR)
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
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
```

Użycie notatek

1. Funkcja wykonywana przez wyjście kanału jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami zdefiniowanymi w tym miejscu i w powiązanym bloku kontrolnym MQCXP.
2. Parametr **ChannelDefinition** przekazany do wyjścia kanału może mieć jedną z kilku wersji. Więcej informacji na ten temat zawiera pole *Version* w strukturze MQCD.
3. Jeśli wyjście kanału odbiera strukturę MQCD z polem *Version* ustawionym na wartość większą niż MQCD_VERSION_1, wyjście musi używać pola *ConnectionName* w MQCD, a nie pola *ShortConnectionName* .
4. Ogólnie rzecz biorąc, wyjścia kanału mogą zmieniać długość danych komunikatu. Może to wynikać z wyjścia dodającego dane do komunikatu lub usuwającego dane z komunikatu albo kompresującego lub szyfrującego komunikat. Jeśli jednak komunikat jest segmentem zawierającym tylko część

komunikatu logicznego, obowiązują specjalne ograniczenia. W szczególności nie może występować zmiana netto długości komunikatu w wyniku działań uzupełniających wyjścia nadawcze i odbiorcze.

Na przykład dozwolone jest, aby wyjście wysyłania skróciło komunikat poprzez jego kompresję, ale dodatkowe wyjście odbierania musi przywrócić oryginalną długość komunikatu poprzez jego dekompresję, tak aby nie doszło do zmiany długości komunikatu.

To ograniczenie wynika z faktu, że zmiana długości segmentu spowodowałaby, że przesunięcia późniejszych segmentów w komunikacie byłyby niepoprawne, co uniemożliwiłoby menedżerowi kolejek rozpoznanie, że segmenty utworzyły pełny komunikat logiczny.

MQ_CHANNEL_AUTO_DEF_EXIT-wyjście automatycznej definicji kanału

Wywołanie MQ_CHANNEL_AUTO_DEF_EXIT opisuje parametry, które są przekazywane do wyjścia automatycznej definicji kanału wywoływanego przez agenta kanału komunikatów.

Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQ_CHANNEL_AUTO_DEF_EXIT. Nazwa MQ_CHANNEL_AUTO_DEF_EXIT nie ma specjalnego znaczenia, ponieważ w menedżerze kolejek udostępniono nazwy wyjść automatycznego definiowania.

Składnia

MQ_CHANNEL_AUTO_DEF_EXIT *(ChannelExitParms, ChannelDefinition)*

Parametry

Wywołanie MQ_CHANNEL_AUTO_DEF_EXIT ma następujące parametry.

ChannelExit-parametry wejściowe/wyjściowe (MQCXP)

Blok parametru wyjścia kanału.

Ta struktura zawiera dodatkowe informacje dotyczące wywołania wyjścia.Wyjście ustawia informacje w tej strukturze w celu wskazania sposobu działania agenta MCA.

ChannelDefinition (MQCD)-wejście/wyjście

Definicja kanału.

Ta struktura zawiera parametry ustawione przez administratora w celu sterowania zachowaniem kanałów, które są tworzone automatycznie.Wyjście ustawia informacje w tej strukturze w celu zmodyfikowania domyślnego zachowania ustawionego przez administratora.

Wymienione pola MQCD nie mogą być zmieniane przez wyjście:

- *ChannelName*
- *ChannelType*
- *StrucLength*
- *Version*

Jeśli inne pola zostaną zmienione, wartość ustawiona przez wyjście musi być poprawna. Jeśli wartość nie jest poprawna, komunikat o błędzie jest zapisywany w pliku dziennika błędów lub wyświetlany na konsoli (w zależności od środowiska).

Ostrzeżenie: Automatycznie definiowane kanały utworzone przez wyjście automatycznej definicji kanału (CHAD) nie mogą ustawić etykiety certyfikatu, ponieważ uzgadnianie TLS miało miejsce przed utworzeniem kanału. Ustawienie etykiety certyfikatu w wyjściu CHAD dla kanałów przychodzących nie ma wpływu.

Wywołanie C

```
exitname (&ChannelExitParms, &ChannelDefinition);
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
MQCXP  ChannelExitParms;   /* Channel exit parameter block */
MQCD   ChannelDefinition;  /* Channel definition */
```

Wywołanie COBOL

```
     CALL 'exitname' USING CHANNELEXITPARMS, CHANNELDEFINITION.
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
**   Channel exit parameter block
 01  CHANNELEXITPARMS.
     COPY CMQCXPV.
**   Channel definition
 01  CHANNELDEFINITION.
     COPY CMQCDV.
```

Wywołanie RPG (ILE)

```
     C*..1....:....2....:....3....:....4....:....5....:....6....:....7..
     C                    CALLP     exitname(MQCXP : MQCD)
```

Definicja prototypu dla wywołania jest następująca:

```
     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
```

Wywołanie asemblera systemu System/390

```
         CALL EXITNAME,(CHANNELEXITPARMS,CHANNELDEFINITION)
```

Parametry przekazywane do wyjścia są deklarowane w następujący sposób:

```
CHANNELEXITPARMS   CMQCXPA  ,  Channel exit parameter block
CHANNELDEFINITION  CMQCDA   ,  Channel definition
```

Użycie notatek

1. Funkcja wykonywana przez wyjście kanału jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami zdefiniowanymi w tym miejscu i w powiązanym bloku kontrolnym MQCXP.
2. Parametr **ChannelExitParms** przekazany do wyjścia automatycznej definicji kanału jest strukturą MQCXP. Przekazana wersja produktu MQCXP zależy od środowiska, w którym działa wyjście. Szczegółowe informacje można znaleźć w opisie pola *Version* w pliku “MQCXP-parametr wyjścia kanału” na stronie 1568 .
3. Parametr **ChannelDefinition** przekazany do wyjścia automatycznej definicji kanału jest strukturą MQCD. Przekazana wersja produktu MQCD zależy od środowiska, w którym działa wyjście. Szczegółowe informacje można znaleźć w opisie pola *Version* w pliku “MQCD-definicja kanału” na stronie 1526 .

MQXWAIT-oczekiwanie na wyjście

Wywołanie MQXWAIT oczekuje na wystąpienie zdarzenia. Można go używać tylko z poziomu wyjścia kanału w systemie z/OS.

Użycie komendy MQXWAIT pomaga uniknąć problemów z wydajnością, które mogą wystąpić w przeciwnym razie, jeśli wyjście kanału wykona działanie powodujące oczekiwanie. Zdarzenie, na które oczekuje MQXWAIT, jest sygnalizowane przez system MVS ECB (blok kontrolny zdarzenia). EBC jest opisany w opisie bloku kontrolnego MQXWD.

z/OS Więcej informacji na temat używania funkcji MQXWAIT i zapisywania programów obsługi wyjścia kanału zawiera sekcja Zapisywanie programów obsługi wyjścia kanału w systemie z/OS

Składnia

OCZEKIWANIE MQXWAIT ***(Hconn, WaitDesc, CompCode, Reason)***

Parametry

Wywołanie MQXWAIT ma następujące parametry.

Hconn (MQHCONN)-wejście

Uchwyt połączenia.

Ten uchwyt reprezentuje połączenie z menedżerem kolejek. Wartość *Hconn* została zwrócona przez poprzednie wywołanie MQCONN wydane w tym samym lub wcześniejszym wywołaniu wyjścia.

WaitDesc (MQXWD)-wejście/wyjście

Deskryptor oczekiwania.

Ten parametr opisuje zdarzenie, na które należy czekać. Szczegółowe informacje na temat pól w tej strukturze zawiera sekcja “MQXWD-Wyjście z deskryptora oczekiwania” na stronie 1583 .

CompCode (MQLONG)-dane wyjściowe

Kod zakończenia.

Jest to jeden z następujących kodów:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe

Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

MQRC_ADAPTER_NIEDOSTĘPNE
(2204, X'89C') Adapter nie jest dostępny.

BŁĄD MQRC_OPTIONS_ERROR
(2046, X'7FE') Opcje są niepoprawne lub niespójne.

MQRC_XWAIT_ANULOWANA
(2107, X'83B') Wywołanie MQXWAIT zostało anulowane.

BŁĄD MQRC_XWAIT_ERROR
(2108, X'83C') Niepoprawne wywołanie wywołania MQXWAIT.

Wywołanie C

```
MQXWAIT (Hconn, &WaitDesc, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONN  Hconn;     /* Connection handle */
MQXWD    WaitDesc;  /* Wait descriptor */
MQLONG   CompCode;  /* Completion code */
MQLONG   Reason;    /* Reason code qualifying CompCode */
```

Wywołanie asemblera systemu System/390

```
         CALL MQXWAIT,(HCONN,WAITDESC,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
HCONN     DS       F  Connection handle
WAITDESC  CMQXWDA  ,  Wait descriptor
COMPCODE  DS       F  Completion code
REASON    DS       F  Reason code qualifying COMPCODE
```

MQCD-definicja kanału

Struktura MQCD zawiera parametry sterujące wykonywaniem kanału. Jest on przekazywany do każdego wyjścia kanału wywoływanego z agenta kanału komunikatów (MCA).

Więcej informacji na temat wyjść kanałów zawiera sekcja “MQ_CHANNEL_EXIT-wyjście kanału” na stronie 1519. Opis w tym temacie dotyczy zarówno kanałów komunikatów, jak i kanałów MQI.

Pola nazwy wyjścia

Po wywołaniu wyjścia odpowiednie pole z *SecurityExit*, *MsgExit*, *SendExit*, *ReceiveExit*i *MsgRetryExit* zawiera nazwę aktualnie wywoływanego wyjścia. Znaczenie nazwy w tych polach zależy od środowiska, w którym działa agent MCA. Z wyjątkiem przypadków, gdy zaznaczono inaczej, nazwa jest wyrównywana do lewej strony pola bez odstępów wewnętrznych; nazwa jest uzupełniana odstępami do długości pola. W poniższych opisach nawiasy kwadratowe ([]) oznaczają informacje opcjonalne:

AIX and Linux
Nazwa wyjścia jest nazwą dynamicznie ładowanego modułu lub biblioteki z przyrostkiem nazwy funkcji rezydującej w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwa biblioteki może być opcjonalnie poprzedzona ścieżką do katalogu:

```
[ path ] library ( function )
```

Długość nazwy nie może przekraczać 128 znaków.

z/OS
Nazwa wyjścia jest nazwą modułu ładującego, która jest poprawna dla specyfikacji w parametrze EP makra LINK lub LOAD. Długość nazwy nie może przekraczać ośmiu znaków.

Windows
Nazwa wyjścia jest nazwą biblioteki dołączanej dynamicznie, z dołączonym przyrostkiem nazwy funkcji rezydującej w tej bibliotece. Nazwa funkcji musi być ujęta w nawiasy. Nazwę biblioteki można opcjonalnie poprzedzić ścieżką do katalogu i napędem:

```
[d:][ path ] library ( function )
```

Długość nazwy nie może przekraczać 128 znaków.

IBM i

Nazwa wyjścia jest 10-bajtową nazwą programu, po której następuje 10-bajtowa nazwa biblioteki. Jeśli nazwy są krótsze niż 10 bajtów, każda nazwa jest uzupełniana odstępami w celu uzyskania 10 bajtów. Nazwą biblioteki może być `*LIBL` , z wyjątkiem wywoływania wyjścia automatycznej definicji kanału, w którym to przypadku wymagana jest pełna nazwa.

Zmiana pól MQCD w wyjściu kanału

Wyjście kanału może zmieniać pola w MQCD. Zmieniona wartość pozostaje w MQCD i jest przekazywana do wszystkich pozostałych wyjść w łańcuchu wyjścia i do każdej konwersacji współużytkującej instancję kanału. Zmieniony MQCD jest również używany przez agent MCA na potrzeby normalnego przetwarzania w trakcie ciągłego czasu życia kanału.

Następujące pola MQCD nie mogą być zmieniane przez wyjście:

- ChannelName
- ChannelType
- StrucLength
- Wersja

Odsyłacze pokrewne

“Pola” na stronie 1527
W tym temacie opisano wszystkie pola w strukturze MQCD oraz opisano poszczególne pola.

“Deklaracja C” na stronie 1555
Ta deklaracja jest deklaracją C dla struktury MQCD.

“Deklaracja języka COBOL” na stronie 1557
Ta deklaracja jest deklaracją języka COBOL dla struktury MQCD.

“Deklaracja RPG (ILE)” na stronie 1559
Ta deklaracja jest deklaracją języka RPG dla struktury MQCD.

“Deklaracja asemblera System/390” na stronie 1562
Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQCD.

“Deklaracja Visual Basic” na stronie 1563
Ta deklaracja jest deklaracją Visual Basic struktury MQCD.

“Zmiana pól MQCD w wyjściu kanału” na stronie 1565
Wyjście kanału może zmieniać pola w MQCD. Jednak zmiany te zwykle nie są wykonywane, z wyjątkiem wymienionych okoliczności.

Pola

W tym temacie opisano wszystkie pola w strukturze MQCD oraz opisano poszczególne pola.

Limit BatchData(MQLONG)
To pole określa limit (w kilobajtach) ilości danych, które mogą być wysyłane przez kanał przed przejściem do punktu synchronizacji.

Punkt synchronizacji jest pobierany po przejściu przez kanał komunikatu, który spowodował osiągnięcie limitu.

Zadanie wsadowe jest przerywane, gdy spełniony zostaje jeden z następujących warunków:

- Liczba wysłanych komunikatów: **`BatchSize`** .
- Liczba wysłanych bajtów: **`BatchDataLimit`** .
- Kolejka transmisji jest pusta i została przekroczona wartość **`BatchInterval`** .

Wartość musi być z zakresu od 0 do 999999. Wartość domyślna to 5000.

Wartość zero w tym atrybucie oznacza, że do zadań wsadowych w tym kanale nie jest stosowany żaden limit danych.

Ten parametr ma zastosowanie tylko do kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSRCVR lub MQCHT_CLUSSDR.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_11.

BatchHeartbeat (MQLONG)
To pole określa przedział czasu, który jest używany do wyzwalania pulsu przetwarzania wsadowego dla kanału.

Puls wsadowy umożliwia kanałom nadawczym określenie, czy instancja kanału zdalnego jest nadal aktywna przed przejściem w stan wątpliwy. Puls zadania wsadowego występuje, jeśli kanał nadawczy nie skomunikował się z instancją kanału zdalnego w określonym przedziale czasu.

Wartość należy do zakresu od 0 do 999 999; jednostkami są milisekundy. Wartość zero oznacza, że puls zadania wsadowego nie jest włączony.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

BatchInterval (MQLONG)
To pole określa przybliżony czas (w milisekundach), przez który kanał utrzymuje otwarte zadanie wsadowe, jeśli w bieżącym zadaniu wsadowym przesłano mniej niż *BatchSize* komunikatów.

Jeśli wartość parametru *BatchInterval* jest większa od zera, zadanie wsadowe jest przerywane przez pierwsze z następujących zdarzeń:

- Liczba wysłanych komunikatów: *BatchSize* lub
- Liczba milisekund, które upłynęły od uruchomienia zadania wsadowego: *BatchInterval* .

Jeśli parametr *BatchInterval* ma wartość zero, zadanie wsadowe jest przerywane przez pierwsze z następujących zdarzeń:

- Liczba wysłanych komunikatów: *BatchSize* lub
- kolejka transmisji staje się pusta.

Wartość *BatchInterval* musi należeć do zakresu od 0 do 999 999 999.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to pole wejściowe do wyjścia. To pole nie jest obecne, gdy *Version* jest mniejsze niż MQCD_VERSION_4.

BatchSize (MQLONG)
To pole określa maksymalną liczbę komunikatów, które mogą być wysyłane przez kanał przed synchronizacją kanału.

To pole nie dotyczy kanałów z wartością *ChannelType* wynoszącą MQCHT_SVRCONN lub MQCHT_CLNTCONN.

CertificateLabel (MQCHAR64)
W tym polu znajdują się szczegółowe informacje na temat używanej etykiety certyfikatu.

IBM MQ inicjuje domyślną wartość w polu *CertificateLabel* jako pustą.

Jest on interpretowany w czasie wykonywania jako wartość domyślna i jest kompatybilny wstecz.

Na przykład podanie wersji MQCD mniejszej niż 11 lub użycie domyślnej wartości odstępu w polu *CertificateLabel* oznacza, że to pole jest ignorowane.

Długość tego pola jest określona przez wartość MQ_CERT_LABEL_LENGTH.

ChannelMonitoring (MQLONG)
To pole określa bieżący poziom gromadzenia danych monitorowania dla kanału.

To pole nie dotyczy kanałów, dla których ChannelType ma wartość MQCHT_CLNTCONN.

Jest to jedna z następujących wartości:

- MQMON_WYŁ
- MQMON_NISKI
- MQMON_MEDIUM
- MQMON_HIGH

Jest to pole wejściowe do wyjścia. Nie jest ona obecna, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

ChannelName (MQCHAR20)
To pole określa nazwę definicji kanału.

Aby możliwa była komunikacja, na komputerze zdalnym musi istnieć definicja kanału o takiej samej nazwie.

Nazwa może zawierać tylko następujące znaki:

- Wielkie litery A-Z
- Małe litery a-z
- Cyfry 0-9
- Kropka (.)
- Prawy ukośnik (/)
- Podkreślenie (_)
- Znak procentu (%)

i być dopełniane odstępami po prawej stronie. Czołowe lub wewnętrzne odstępy nie są dozwolone.

Długość tego pola jest określona przez wartość MQ_CHANNEL_NAME_LENGTH.

ChannelStatistics (MQLONG)
To pole określa bieżący poziom gromadzenia danych statystycznych dla kanału.

To pole nie dotyczy kanałów, dla których ChannelType ma wartość MQCHT_CLNTCONN lub MQCHT_SVRCONN.

Jest to jedna z następujących wartości:

- MQMON_WYŁ
- MQMON_NISKI
- MQMON_MEDIUM
- MQMON_HIGH

Jest to pole wejściowe do wyjścia. Nie jest ona obecna, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

ChannelType (MQLONG)
To pole określa typ kanału.

Jest to jedna z następujących wartości:

MQCHT_SENDER
Nadawca.

SERWER_MQ
Serwer.

MQCHT_RECEIVER
Dziennik.

MQCHT_REQUESTER
Requester.

MQCHT_CLNTCONN
Połączenie klienta.

MQCHT_SVRCONN
Połączenie z serwerem (do użytku przez klientów).

MQCHT_CLUSSDR
Nadawca klastra.

MQCHT_CLUSRCVR
Odbiornik klastra.

ClientChannelWaga (MQLONG)
To pole określa wagę wpływającą na używaną definicję kanału połączenia klienckiego.

Atrybut wagi ClientChannelsłuży do losowego wybierania definicji kanału klienta na podstawie ich wagi, jeśli dostępna jest więcej niż jedna odpowiednia definicja. Gdy klient wysyła żądanie połączenia MQCONN z grupą menedżera kolejek, podając nazwę menedżera kolejek rozpoczynającą się gwiazdką i gdy w tabeli definicji kanału klienta (CCDT) dostępna jest więcej niż jedna odpowiednia definicja kanału, definicja do użycia jest wybierana losowo na podstawie wagi, a wszystkie odpowiednie definicje ClientChannel(0) są wybierane jako pierwsze w kolejności alfabetycznej.

Określ wartość z zakresu od 0 do 99. Wartość domyślna to 0.

Wartość 0 wskazuje brak równoważenia obciążenia, a odpowiednie definicje są wybierane w porządku alfabetycznym. Aby włączyć równoważenie obciążenia, wybierz wartość z zakresu od 1 do 99, gdzie 1 to najniższa waga, a 99 to najwyższa waga. Rozkład komunikatów między dwoma lub większą liczbą kanałów z niezerowymi wagami jest proporcjonalny do stosunku tych wag. Na przykład trzy kanały z wartościami wagi ClientChannelwynoszącymi 2, 4 i 14 są wybierane w przybliżeniu 10%, 20% i 70% czasu. Ta dystrybucja nie jest gwarantowana.

Ten atrybut jest poprawny tylko dla typu kanału połączenia klienckiego.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość w polu *Wersja* jest mniejsza niż MQCD_VERSION_9.

ClusterPtr (MQPTR)
To pole określa adres listy nazw klastrów.

Jeśli wartość *ClustersDefined* jest większa od zera, jest to adres listy nazw klastrów. Kanał należy do każdego wymienionego klastra.

To pole dotyczy tylko kanałów z *ChannelType* typu MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

ClustersDefined (MQLONG)
To pole określa liczbę klastrów, do których należy kanał.

W tym polu znajduje się liczba nazw klastrów wskazywana przez zmienną *ClusterPtr*. Wartość ta wynosi zero lub więcej.

To pole dotyczy tylko kanałów z *ChannelType* typu MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

CLWLChannelPriority (MQLONG)
To pole określa priorytet kanału obciążenia klastra.

Algorytm wyboru menedżera obciążenia wybiera miejsce docelowe o najwyższym priorytecie z zestawu miejsc docelowych wybranych na podstawie rankingu. Jeśli istnieją dwa możliwe menedżery kolejek docelowych, tego atrybutu można użyć do przełączenia awaryjnego jednego menedżera kolejek na inny menedżer kolejek. Wszystkie komunikaty są kierowane do menedżera kolejek z najwyższym priorytetem do momentu zakończenia, a następnie do menedżera kolejek z następnym najwyższym priorytetem.

Wartość należy do zakresu od 0 do 9. Wartość domyślna to 0.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

Więcej informacji na ten temat zawiera sekcja Konfigurowanie klastra menedżera kolejek.

CLWLChannelRank (MQLONG)
To pole określa klasyfikację kanału obciążenia klastra.

Algorytm wyboru menedżera obciążenia wybiera miejsce docelowe o najwyższym stopniu klasyfikacji. Jeśli ostatecznym miejscem docelowym jest menedżer kolejek w innym klastrze, można ustawić pozycję pośrednich menedżerów kolejek bramy (na przecięciu sąsiednich klastrów), tak aby algorytm wyboru poprawnie wybierał docelowy menedżer kolejek bliżej miejsca docelowego.

Wartość należy do zakresu od 0 do 9. Wartość domyślna to 0.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

Więcej informacji na ten temat zawiera sekcja Konfigurowanie klastra menedżera kolejek.

CLWLChannelWeight (MQLONG)
To pole określa wagę kanału obciążenia klastra.

Waga kanału obciążenia klastra.

Algorytm wyboru menedżera obciążenia używa atrybutu "weight" kanału do zniekształcenia wyboru miejsca docelowego, aby można było wysłać więcej komunikatów do konkretnego komputera. Na przykład można nadać kanałowi na dużym serwerze UNIX większą "wagę" niż innemu kanałowi na małym komputerze PC, a algorytm wyboru wybierze serwer UNIX częściej niż komputer PC.

Wartość należy do zakresu od 1 do 99. Domyślną wartością jest 50.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

Więcej informacji na ten temat zawiera sekcja Konfigurowanie klastra menedżera kolejek.

ConnectionAffinity (MQLONG)
To pole określa, czy aplikacje klienckie, które łączą się wiele razy przy użyciu tej samej nazwy menedżera kolejek, używają tego samego kanału klienta.

Ten atrybut jest używany, jeśli dostępnych jest wiele definicji kanałów.

Jest to jedna z następujących wartości:

MQCAFTY_PREFERRED
Pierwsze połączenie w procesie odczytującym tabelę definicji kanału klienta (CCDT) tworzy listę odpowiednich definicji na podstawie wagi z odpowiednimi definicjami CLNTWGHT (0) w pierwszej kolejności i w kolejności alfabetycznej. Każde połączenie w procesie próbuje nawiązać połączenie przy użyciu pierwszej definicji z listy. Jeśli nawiązanie połączenia nie powiedzie się, używana jest następna definicja. Definicje zakończone niepowodzeniem z wartościami CLNTWGHT innymi niż 0 są przenoszone na koniec listy. Definicje CLNTWGHT(0) pozostają na początku listy i są wybierane w pierwszej kolejności przy każdym nawiązywaniu połączenia.

Każdy proces klienta o takiej samej nazwie hosta zawsze tworzy tę samą listę.

W przypadku aplikacji klienckich napisanych w języku C, C + + lub w .NET środowisku programistycznym (w tym w pełni zarządzanym .NET) lista jest aktualizowana, jeśli tabela CCDT została zmodyfikowana od momentu utworzenia listy.

Jest to wartość domyślna.

MQCAFTY_NONE
Pierwsze połączenie w procesie odczytu CCDT tworzy listę odpowiednich definicji. Wszystkie połączenia w procesie wybierają odpowiednią definicję w oparciu o wagę każdej odpowiedniej definicji CLNTWGHT(0) wybranej najpierw zgodnie z porządkiem alfabetycznym.

W przypadku aplikacji klienckich napisanych w języku C, C + + lub w .NET środowisku programistycznym (w tym w pełni zarządzanym .NET) lista jest aktualizowana, jeśli tabela CCDT została zmodyfikowana od momentu utworzenia listy.

Ten atrybut jest poprawny tylko dla typu kanału połączenia klienckiego.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość w polu *Wersja* jest mniejsza niż MQCD_VERSION_9.

ConnectionName (MQCHAR264)
To pole określa nazwę połączenia dla kanału.

W przypadku kanałów odbierających klastry (jeśli określono) parametr CONNAME odnosi się do lokalnego menedżera kolejek, a w przypadku innych kanałów odnosi się do docelowego menedżera kolejek. Podana wartość zależy od używanego protokołu transmisji (*TransportType*):

- Dla MQXPT_LU62jest to pełna nazwa partnerskiej jednostki logicznej.
- Dla MQXPT_NETBIOS jest to nazwa NetBIOS zdefiniowana na komputerze zdalnym.
- W przypadku protokołu MQXPT_TCP jest to nazwa hosta, adres sieciowy komputera zdalnego określony w formacie IPv4 dziesiętnym z kropkami lub IPv6 szesnastkowym lub komputer lokalny dla kanałów odbiorczych klastra.
- Dla MQXPT_SPX jest to adres w stylu SPX składający się z 4-bajtowego adresu sieciowego, 6-bajtowego adresu węzła i 2-bajtowego numeru gniazda.

Podczas definiowania kanału to pole nie ma zastosowania w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_SVRCONN lub MQCHT_RECEIVER. Jeśli jednak definicja kanału jest przekazywana do wyjścia, to pole zawiera adres partnera, niezależnie od typu kanału.

Długość tego pola jest określona przez wartość MQ_CONN_NAME_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

DataConversion (MQLONG)
To pole określa, czy agent kanału komunikatów wysyłających próbuje dokonać konwersji danych komunikatu aplikacji, jeśli agent kanału komunikatów odbierających nie może wykonać tej konwersji.

To pole ma zastosowanie tylko do komunikatów, które nie są segmentami komunikatów logicznych; agent MCA nigdy nie próbuje przekształcić komunikatów, które są segmentami.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR. Jest to jedna z poniższych nazw:

KONWERSJI MQCDC_SENDER_CONVERSION
Konwersja według nadawcy.

MQCDC_NO_SENDER_CONVERSION,
Brak konwersji przez nadawcę.

DefReconnect (MQLONG)
Atrybut kanału `DefReconnect` ustawia domyślną wartość atrybutu ponownego połączenia dla kanału połączenia klienta.

Domyślna opcja automatycznego ponownego nawiązywania połączenia z klientem. Produkt IBM MQ MQI client można skonfigurować w taki sposób, aby automatycznie ponownie łączył się z aplikacją kliencką. Klient IBM MQ MQI client podejmuje próbę ponownego nawiązania połączenia z menedżerem kolejek po niepowodzeniu połączenia. Podejmowana jest próba ponownego nawiązania połączenia bez wysyłania wywołania MQI MQCONN lub MQCONNX przez klient aplikacji.

Ponowne połączenie jest opcją MQCONNX . Za pomocą atrybutu kanału DefReconnect można dodać zachowanie związane z ponownym nawiązaniem połączenia do istniejących aplikacji, które korzystają z produktu MQCONN. Można również zmienić zachowanie ponownego połączenia aplikacji, które korzystają z produktu MQCONNX.

Można również ustawić wartość DefRecon w pliku mqclient.ini , aby ustawić lub zmodyfikować zachowanie ponownego połączenia. Wartość DefRecon z pliku mqclient.ini ma pierwszeństwo przed atrybutem kanału DefReconnect .

Syntax

DefReconnect (MQRCN_NO (default) |MQRCN_YES|MQRCN_Q_MGR|
MQRCN_DISABLED)

Parametry

MQRCN_NO
MQRCN_NO to wartość domyślna.

Jeśli nie zostanie przesłonięte przez **MQCONNX**, klient nie zostanie automatycznie ponownie połączony.

MQRCN_YES
Jeśli nie zostanie przesłonięte przez **MQCONNX**, klient automatycznie nawiąże ponowne połączenie.

MQRCN_Q_MGR
Jeśli nie zostanie przesłonięte przez parametr **MQCONNX**, klient automatycznie ponownie nawiązuje połączenie, ale tylko z tym samym menedżerem kolejek. Opcja QMGR działa tak samo jak opcja MQCNO_RECONNECT_Q_MGR.

MQRCN_DISABLED
Ponowne połączenie jest wyłączone, nawet jeśli program kliencki zażądał ponownego połączenia za pomocą wywołania MQI produktu **MQCONNX** .

Automatyczne ponowne nawiązywanie połączenia przez klient nie jest obsługiwane przez produkt IBM MQ classes for Java.

Tabela 822. Automatyczne ponowne połączenie zależy od wartości ustawionych w aplikacji i definicji kanału

DefReconnect	**Opcje ponownego połączenia ustawione w aplikacji**			
	MQCNO_RECONNECT	MQCNO_RECONNECT_Q_MGR	MQCNO_RECONNECT_AS_DEF	MQCNO_RECONNECT_DISABLED
MQRCN_NO	YES	QMGR	NO	NO
MQRCN_YES	YES	QMGR	YES	NO
MQRCN_Q_MGR	YES	QMGR	QMGR	NO
MQRCN_DISABLED	NO	NO	NO	NO

Pojęcia pokrewne
Automatyczne ponowne łączenie klienta
Ponowne połączenie kanału i klienta
Sekcja CHANNELS pliku konfiguracyjnego klienta

Odsyłacze pokrewne

"Opcje (MQLONG) dla MQCNO" na stronie 327
Opcje sterujące działaniem komendy MQCONNX.

Opis (MQCHAR64)
To pole może być używane jako komentarz opisowy.

Zawartość pola nie ma znaczenia dla agentów kanału komunikatów. Musi on jednak zawierać tylko znaki, które mogą być wyświetlane. Nie może zawierać żadnych znaków null; w razie potrzeby jest dopełniana do prawej strony odstępami. W instalacji DBCS pole może zawierać znaki DBCS (z maksymalną długością pola wynoszącą 64 bajty).

Uwaga: Jeśli to pole zawiera znaki, które nie znajdują się w zestawie znaków menedżera kolejek (zdefiniowanym w atrybucie menedżera kolejek **CodedCharSetId**), znaki te mogą być niepoprawnie przetłumaczone, jeśli zostanie ono wysłane do innego menedżera kolejek.

Długość tego pola jest określona przez wartość MQ_CHANNEL_DESC_LENGTH.

DiscInterval (MQLONG)
To pole określa maksymalny czas (w sekundach), przez który kanał oczekuje na przybycie komunikatu do kolejki transmisji przed zakończeniem kanału.

Innymi słowy, określa odstęp czasu między rozłączeniami.

Wartość A równa zero powoduje, że agent MCA oczekuje w nieskończoność.

W przypadku kanałów połączenia z serwerem korzystających z protokołu TCP odstęp czasu reprezentuje wartość rozłączania nieaktywności klienta określoną w sekundach. Jeśli w tym czasie połączenie z serwerem nie otrzymało żadnej komunikacji od swojego klienta partnerskiego, przerywa ono połączenie. Odstęp czasu nieaktywności połączenia z serwerem ma zastosowanie tylko między wywołaniami API IBM MQ z klienta, dlatego żaden klient nie jest rozłączany podczas długotrwałego wywołania MQGET z oczekiwaniem.

Ten atrybut nie ma zastosowania w przypadku kanałów połączenia z serwerem korzystających z protokołów innych niż TCP.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR, MQCHT_CLUSRCVR lub MQCHT_SVRCONN.

ExitDataDługość (MQLONG)
To pole określa długość (w bajtach) każdego elementu danych użytkownika na liście elementów danych użytkownika programu zewnętrznego adresowanych przez pola *MsgUserDataPtr*, *SendUserDataPtr*i *ReceiveUserDataPtr* .

Ta długość nie musi być taka sama jak długość danych MQ_EXIT_DATA_LENGTH.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

ExitNameDługość (MQLONG)
To pole określa długość (w bajtach) każdej z nazw na liście nazw wyjść adresowanych przez pola *MsgExitPtr*, *SendExitPtr*i *ReceiveExitPtr* .

Ta długość nie musi być taka sama, jak długość określona w zmiennej MQ_EXIT_NAME_LENGTH.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

HdrCompLista [2] (MQLONG)
To pole określa listę technik kompresji danych nagłówka, które są obsługiwane przez kanał.

Lista zawiera co najmniej jedną z następujących wartości:

MQCOMPRESS_NONE
Dane nagłówka nie są kompresowane.

SYSTEM MQCOMPRESS_SYSTEM
Dane nagłówka są kompresowane.

MQCOMPRESS_NOT_AVAILABLE
Nieużywane wartości na liście są ustawione na tę wartość.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

HeartbeatInterval (MQLONG)
To pole określa czas (w sekundach) między przepływami pulsu.

Interpretacja tego pola zależy od typu kanału:

- Dla kanału typu MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER MQCHT_REQUESTER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR to pole określa czas w sekundach między przepływami pulsu przekazywanymi od wysyłającego agenta MCA, gdy w kolejce transmisji nie ma komunikatów. Dzięki temu odbierający agent MCA będzie mógł wyciszyć kanał. Aby była użyteczna, wartość *HeartbeatInterval* musi być mniejsza niż *DiscInterval*.
- Dla kanału typu MQCHT_CLNTCONN lub MQCHT_SVRCONN z polem konwersacji współużytkowania MQCD ustawionym na zero, to pole jest czasem w sekundach między przepływami pulsu przekazywanymi z agenta MCA serwera, gdy agent MCA wywołał wywołanie MQGET z opcją MQGMO_WAIT w imieniu aplikacji klienckiej. Dzięki temu agent MCA serwera może obsłużyć sytuacje, w których nawiązanie połączenia klienta nie powiedzie się podczas operacji MQGET z opcją MQGMO_WAIT.
- W przypadku kanału typu MQCHT_CLNTCONN lub MQCHT_SVRCONN z polem Współużytkowanie konwersacji MQCD ustawionym na wartość niezerową to pole jest czasem (w sekundach) między przepływami pulsu, gdy nie są wysyłane ani odbierane żadne przepływy danych. Pozwala to na efektywne wyciszenie kanału.

Wartość należy do zakresu od 0 do 999 999. Używana wartość jest większa od wartości podanej po stronie wysyłającej i odbierającej, chyba że po obu stronach zostanie podana wartość 0, co oznacza, że nie występuje wymiana pulsu.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Przedział czasu KeepAlive(MQLONG)
To pole określa wartość przekazywaną do stosu komunikacji dla czasu podtrzymywania połączenia dla kanału.

Wartość ta ma zastosowanie do protokołów komunikacyjnych TCP/IP i SPX, ale nie wszystkie implementacje obsługują ten parametr.

Wartość należy do zakresu od 0 do 99 999; jednostkami są sekundy. Wartość zero wskazuje, że funkcja keepalive kanału nie jest włączona, chociaż funkcja keepalive może nadal występować, jeśli włączona jest funkcja keepalive TCP/IP (a nie keepalive kanału). Poprawna jest także następująca wartość specjalna:

MQKAI_AUTO
Automatyczny.

Ta wartość wskazuje, że interwał sprawdzania połączenia jest obliczany na podstawie wynegocjowanego interwału pulsu w następujący sposób:

- Jeśli wynegocjowany przedział czasu pulsu jest większy od zera, używany jest przedział czasu sprawdzania połączenia (keepalive) powiększony o 60 sekund.
- Jeśli wynegocjowany przedział czasu pulsu wynosi zero, interwał sprawdzania połączenia, który jest używany, wynosi zero.

- W systemie z/OSpodtrzymywanie połączenia TCP/IP występuje, gdy w obiekcie menedżera kolejek określono wartość TCPKEEP (YES).
- W innych środowiskach funkcja podtrzymywania połączenia TCP/IP występuje, gdy parametr **KEEPALIVE=YES** jest określony w sekcji TCP w pliku konfiguracyjnym kolejkowania rozproszonego.

To pole ma zastosowanie tylko w przypadku kanałów, w których parametr *TransportType* ma wartość MQXPT_TCP lub MQXPT_SPX.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

LocalAddress (MQCHAR48)
To pole określa lokalny adres TCP/IP zdefiniowany dla kanału komunikacji wychodzącej.

To pole jest puste, jeśli nie zdefiniowano konkretnego adresu dla komunikacji wychodzącej. Adres może opcjonalnie zawierać numer portu lub zakres numerów portów. Format tego adresu jest następujący:

```
[ip-addr][(low-port[,high-port])]
```

gdzie nawiasy kwadratowe ([]) oznaczają informacje opcjonalne, `ip-addr` jest podane w IPv4 postaci dziesiętnej z kropkami, IPv6 szesnastkowej lub alfanumerycznej, a `low-port` i `high-port` są numerami portów ujętymi w nawiasy. Wszystkie są opcjonalne.

Konkretny adres IP, port lub zakres portów dla komunikacji wychodzącej jest przydatny w scenariuszach odtwarzania, w których kanał jest restartowany na innym stosie TCP/IP.

Format *LocalAddress* jest podobny do formatu *ConnectionName*, ale nie można go z nim mylić. Parametr *LocalAddress* określa parametry komunikacji lokalnej, natomiast parametr *ConnectionName* określa sposób nawiązania połączenia ze zdalnym menedżerem kolejek.

W produkcie IBM MQ 9.3.0zaktualizowano interfejs JMQI (Message Queueing Interface) produktu Java , aby zapewnić ustawienie pola adresu lokalnego w obiekcie MQCD po utworzeniu instancji kanału i nawiązaniu połączenia z menedżerem kolejek. Oznacza to, że gdy wyjście kanału zapisane w pliku Java wywołuje metodę `MQCD.getLocalAddress()`, metoda zwraca adres lokalny używany przez instancję kanału. Przed produktem IBM MQ 9.3.0wyjście zabezpieczeń kanału nie mogło uzyskać dostępu do adresu lokalnego używanego przez instancję kanału, a metoda `MQCD.getLocalAddress()` zwróciła wartość NULL.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *TransportType* ma wartość MQXPT_TCP, a parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLNTCONN, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Długość tego pola jest określona przez wartość MQ_LOCAL_ADDRESS_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

LongMCAUserIdLength (MQLONG)
To pole określa długość (w bajtach) pełnego identyfikatora użytkownika MCA określonego przez *LongMCAUserIdPtr*.

To pole nie dotyczy kanałów z wartością *ChannelType* wynoszącą MQCHT_CLNTCONN.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

LongMCAUserIdPtr (MQPTR)
To pole określa adres długiego identyfikatora użytkownika MCA.

Jeśli wartość *LongMCAUserIdLength* jest większa od zera, to pole zawiera adres pełnego identyfikatora użytkownika MCA. Długość pełnego identyfikatora jest określona przez parametr *LongMCAUserIdLength*. Pierwsze 12 bajtów identyfikatora użytkownika MCA również znajduje się w polu *MCAUserIdentifier*.

Szczegółowe informacje na temat identyfikatora użytkownika MCA znajdują się w opisie pola *MCAUserIdentifier* .

To pole nie dotyczy kanałów, których parametr *ChannelType* ma wartość MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN lub MQCHT_CLUSSDR.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

LongRemoteUserIdLength (MQLONG)
To pole określa długość (w bajtach) pełnego identyfikatora użytkownika zdalnego wskazywanego przez *LongRemoteUserIdPtr*.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_CLNTCONN lub MQCHT_SVRCONN.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

LongRemoteUserIdPtr (MQPTR)
Pole to określa adres długiego identyfikatora użytkownika zdalnego.

Jeśli wartość *LongRemoteUserIdLength* jest większa od zera, ta opcja jest adresem pełnego identyfikatora użytkownika zdalnego. Długość pełnego identyfikatora jest określona przez parametr *LongRemoteUserIdLength*. Pierwsze 12 bajtów identyfikatora użytkownika zdalnego znajduje się również w polu *RemoteUserIdentifier*.

Szczegółowe informacje na temat identyfikatora użytkownika zdalnego można znaleźć w opisie pola *RemoteUserIdentifier* .

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_CLNTCONN lub MQCHT_SVRCONN.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Liczba LongRetry(MQLONG)
To pole określa liczbę używaną po wyczerpaniu liczby określonej przez zmienną *ShortRetryCount* .

Określa maksymalną liczbę dalszych prób nawiązania połączenia z komputerem zdalnym, w odstępach czasu określonych przez parametr *LongRetryInterval*, przed zarejestrowaniem błędu w operatorze.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Odstęp czasu LongRetry(MQLONG)
To pole określa maksymalną liczbę sekund oczekiwania przed ponowną próbą nawiązania połączenia z komputerem zdalnym.

Odstęp czasu między ponownymi próbami można wydłużyć, jeśli kanał ma oczekiwać na aktywowanie.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

MaxInstances (MQLONG)
To pole określa maksymalną liczbę jednoczesnych instancji pojedynczego kanału połączenia z serwerem, które można uruchomić.

To pole jest używane tylko w kanałach połączenia z serwerem.

Pole może mieć wartość z zakresu od 0 do 999 999 999. Wartość zero uniemożliwia dostęp do klienta.

Wartością domyślną tego pola jest 999 999 999.

Jeśli wartość tego pola zostanie zmniejszona do liczby mniejszej niż liczba instancji kanału połączenia z serwerem, które są obecnie uruchomione, nie będzie to miało wpływu na te działające instancje. Nie można jednak uruchomić nowych instancji, dopóki nie przestanie działać wystarczająca liczba istniejących instancji, tak aby liczba obecnie działających instancji była mniejsza niż wartość w polu.

MaxInstancesPerClient (MQLONG)
To pole określa maksymalną liczbę symultanicznych instancji pojedynczego kanału połączenia z serwerem, które można uruchomić z pojedynczego klienta.

W tym kontekście połączenia pochodzące z tego samego zdalnego adresu sieciowego są traktowane jako przychodzące od tego samego klienta.

To pole jest używane tylko w kanałach połączenia z serwerem.

Pole może mieć wartość z zakresu od 0 do 999 999 999. Wartość zero uniemożliwia dostęp do klienta.

Wartością domyślną tego pola jest 999 999 999.

Jeśli wartość w tym polu zostanie zmniejszona do liczby mniejszej niż liczba instancji kanału połączenia z serwerem, które są obecnie uruchomione z poszczególnych klientów, nie będzie to miało wpływu na te uruchomione instancje. Jednak nowe instancje z dowolnego z tych klientów nie mogą zostać uruchomione, dopóki nie przestanie działać wystarczająca liczba istniejących instancji, tak aby liczba obecnie działających instancji pochodzących od klienta, który próbuje uruchomić nową instancję, była mniejsza niż wartość pola.

MaxMsg(MQLONG)
To pole określa maksymalną długość komunikatu, który może być przesyłany przez kanał.

Jest ona porównywana z wartością kanału zdalnego i z tych dwóch wartości niższą wartością jest bieżąca wartość maksymalna.

Nazwa MCAName (MQCHAR20)
To pole jest polem zarezerwowanym.

Wartość tego pola jest pusta.

Długość tego pola jest określona przez wartość MQ_MCA_NAME_LENGTH.

MCASecurityId (MQBYTE40)
To pole określa identyfikator zabezpieczeń dla agenta MCA.

To pole nie dotyczy kanałów z wartością *ChannelType* wynoszącą MQCHT_CLNTCONN.

Następująca wartość specjalna wskazuje, że nie ma identyfikatora zabezpieczeń:

MQSID_BRAK
Nie określono identyfikatora zabezpieczeń.

Wartością długości pola jest zero binarne.

W przypadku języka programowania C stała MQSID_NONE_ARRAY jest również zdefiniowana. Ta stała ma taką samą wartość jak zmienna MQSID_NONE, ale jest tablicą znaków, a nie łańcuchem.

Jest to pole wejściowe/wyjściowe do wyjścia. Długość tego pola jest określona przez wartość MQ_SECURITY_ID_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Typ MCAType (MQLONG)
To pole określa typ programu agenta kanału komunikatów.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to jedna z następujących wartości:

PROCES_MQMCAT
proces.

Agent kanału komunikatów jest uruchamiany jako oddzielny proces.

WĄTEK MQMCAT_THREAD
Wątek (Wiele platform).

Agent kanału komunikatów jest uruchamiany jako oddzielny wątek.

To pole nie jest wyświetlane, jeśli wartość w polu *Wersja* jest mniejsza niż MQCD_VERSION_2.

MCAUserIdentifier (MQCHAR12)
To pole określa identyfikator użytkownika dla agenta kanału komunikatów (MCA).

To pole używa pierwszych 12 bajtów identyfikatora użytkownika MCA i może zostać ustawione przez agenta zabezpieczeń.

Istnieją dwa pola zawierające identyfikator użytkownika MCA:

- *MCAUserIdentifier* zawiera pierwsze 12 bajtów identyfikatora użytkownika MCA i jest dopełniana odstępami, jeśli identyfikator jest krótszy niż 12 bajtów. Pole *MCAUserIdentifier* może być puste.
- *LongMCAUserIdPtr* wskazuje pełny identyfikator użytkownika MCA, który może być dłuższy niż 12 bajtów. Jego długość jest określona przez parametr *LongMCAUserIdLength*. Pełny identyfikator nie zawiera końcowych odstępów i nie jest zakończony znakiem o kodzie zero. Jeśli identyfikator jest pusty, wartość *LongMCAUserIdLength* wynosi zero, a wartość *LongMCAUserIdPtr* jest niezdefiniowana.

  Uwaga: Parametr *LongMCAUserIdPtr* nie jest obecny, jeśli wartość parametru *Version* jest mniejsza niż MQCD_VERSION_6.

Jeśli identyfikator użytkownika MCA nie jest pusty, określa identyfikator użytkownika, który ma być używany przez agent kanału komunikatów w celu autoryzacji dostępu do zasobów IBM MQ . Dla typów kanałów MQCHT_REQUESTER, MQCHT_RECEIVER i MQCHT_CLUSRCVR, jeśli parametr PutAuthority ma wartość MQPA_DEFAULT, jest to identyfikator użytkownika używany do sprawdzania autoryzacji dla operacji umieszczania w kolejkach docelowych.

Jeśli identyfikator użytkownika MCA jest pusty, agent kanału komunikatów używa domyślnego identyfikatora użytkownika.

Identyfikator użytkownika MCA może zostać ustawiony przez wyjście zabezpieczeń w celu wskazania identyfikatora użytkownika, który musi być używany przez agent kanału komunikatów. Wyjście może zmienić się albo *MCAUserIdentifier*, albo łańcuch wskazywany przez *LongMCAUserIdPtr*. Jeśli obie te wartości zostały zmienione, ale różnią się od siebie, agent MCA korzysta z *LongMCAUserIdPtr* zamiast *MCAUserIdentifier*. Jeśli wyjście zmienia długość łańcucha adresowanego przez *LongMCAUserIdPtr*, należy odpowiednio ustawić *LongMCAUserIdLength* . Jeśli wyjście zwiększa długość identyfikatora, wyjście musi przydzielić pamięć o wymaganej długości, ustawić tę pamięć na wymagany identyfikator i umieścić adres tej pamięci w programie *LongMCAUserIdPtr*. Wyjście jest odpowiedzialne za zwolnienie pamięci masowej, gdy zostanie później wywołane z przyczyną MQXR_TERM.

W przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SVRCONN, jeśli parametr *MCAUserIdentifier* w definicji kanału jest pusty, identyfikator użytkownika przesłany z klienta jest do niego kopiowany. Ten identyfikator użytkownika (po każdej modyfikacji przez wyjście zabezpieczeń na serwerze) jest identyfikatorem, w ramach którego zakłada się, że aplikacja kliencka jest uruchomiona.

Identyfikator użytkownika MCA nie jest odpowiedni dla kanałów z *ChannelType* MQCHT_SDR, MQCHT_SVR, MQCHT_CLNTCONN, MQCHT_CLUSSDR.

Jest to pole wejściowe/wyjściowe do wyjścia. Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

ModeName (MQCHAR8)
To pole określa nazwę trybu jednostki logicznej 6.2 .

To pole ma zastosowanie tylko wtedy, gdy protokołem transmisji (*TransportType*) jest MQXPT_LU62, a *ChannelType* nie jest MQCHT_SVRCONN ani MQCHT_RECEIVER.

To pole jest zawsze puste. Zamiast tego informacje są zawarte w obiekcie pobocznym komunikacji.

Długość tego pola jest określona przez wartość MQ_MODE_NAME_LENGTH.

MsgCompLista [16] (MQLONG)
To pole określa listę technik kompresji danych komunikatu, które są obsługiwane przez kanał.

Lista zawiera co najmniej jedną z następujących wartości:

MQCOMPRESS_NONE
Dane komunikatu nie są kompresowane.

MQCOMPRESS_RLE
Kompresja danych komunikatu jest wykonywana przy użyciu kodowania grupowego.

MQCOMPRESS_ZLIBFAST,
Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowana jest szybka kompresja.

MQCOMPRESS_ZLIBHIGH
Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowany jest wysoki poziom kompresji.

MQCOMPRESS_ANY
Do kompresji komunikatów można użyć dowolnej techniki kompresji obsługiwanej przez menedżer kolejek. Wartość MQCOMPRESS_ANY jest poprawna tylko w przypadku kanałów odbiornika, requestera i połączenia serwera.

MQCOMPRESS_NOT_AVAILABLE
Nieużywane wartości na liście są ustawione na tę wartość.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_8.

MsgExit (MQCHARn)
To pole określa nazwę wyjścia komunikatu kanału.

Jeśli ta nazwa nie jest pusta, wyjście jest wywoływane w następujących momentach:

- Bezpośrednio po pobraniu komunikatu z kolejki transmisji (nadawcy lub serwera) lub bezpośrednio przed umieszczeniem komunikatu w kolejce docelowej (odbiorcy lub requestera).

  Wyjście otrzymuje cały komunikat aplikacji i nagłówek kolejki transmisji do modyfikacji.
- Podczas inicjowania i kończenia kanału.

To pole nie ma zastosowania w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SVRCONN lub MQCHT_CLNTCONN; dla takich kanałów nigdy nie jest wywoływane wyjście komunikatu.

Opis zawartości tego pola w różnych środowiskach zawiera sekcja “MQCD-definicja kanału” na stronie 1526 .

Długość tego pola jest określona przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

MsgExitPtr (MQPTR)
To pole określa adres pierwszego pola *MsgExit* .

Jeśli wartość *MsgExitsDefined* jest większa od zera, ten adres jest adresem listy nazw poszczególnych wyjść komunikatów kanału w łańcuchu.

Każda nazwa znajduje się w polu o długości *ExitNameLength*, dopełnianym do prawej strony odstępami. Istnieją pola *MsgExitsDefined* sąsiadujące ze sobą-po jednym dla każdego wyjścia.

Wszystkie zmiany wprowadzone w tych nazwach przez wyjście są zachowywane, chociaż wyjście kanału komunikatów nie podejmuje jawnego działania-nie zmienia tego, które wyjścia są wywoływane.

Jeśli *MsgExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

MsgExitsZdefiniowane (MQLONG)
To pole określa liczbę wyjść komunikatów kanału zdefiniowanych w łańcuchu.

Jest ona większa lub równa zero.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

MsgRetry(MQLONG)-liczba
To pole określa liczbę prób umieszczenia komunikatu przez agent MCA po pierwszej próbie zakończonej niepowodzeniem.

To pole wskazuje liczbę prób wykonania przez agent MCA operacji otwierania lub umieszczania, jeśli pierwsza operacja MQOPEN lub MQPUT zakończy się niepowodzeniem z kodem zakończenia MQCC_FAILED. Efekt działania tego atrybutu zależy od tego, czy atrybut *MsgRetryExit* jest pusty, czy nie:

- Jeśli parametr *MsgRetryExit* jest pusty, atrybut **MsgRetryCount** określa, czy agent MCA podejmie ponowną próbę. Jeśli wartość atrybutu wynosi zero, nie jest podejmowana żadna ponowna próba. Jeśli wartość atrybutu jest większa od zera, próby ponowień są wykonywane w odstępach czasu określonych przez atrybut **MsgRetryInterval** .

  Ponowienia są wykonywane tylko z następujących kodów przyczyny:

  – MQRC_PAGESET_FULL
  – MQRC_PUT_INHIBITED
  – MQRC_Q_FULL

  W przypadku innych kodów przyczyny agent MCA przechodzi natychmiast do normalnego przetwarzania po awarii bez ponawiania komunikatu o niepowodzeniu.
- Jeśli parametr *MsgRetryExit* ma wartość inną niż pusta, atrybut **MsgRetryCount** nie ma wpływu na agent MCA. Zamiast tego jest to wyjście komunikatu, które określa, ile razy podejmowana jest ponowna próba, i w jakich odstępach czasu. Wyjście jest wywoływane nawet wtedy, gdy atrybut **MsgRetryCount** ma wartość zero.

  Atrybut **MsgRetryCount** jest dostępny dla wyjścia w strukturze MQCD, ale wyjście nie jest wymagane, aby honorować go-ponowne próby będą kontynuowane w nieskończoność, dopóki wyjście nie zwróci wartości MQXCC_SUPPRESS_FUNCTION w polu *ExitResponse* w MQCXP.

To pole ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

Wyjście MsgRetry(MQCHARn)
To pole określa nazwę wyjścia dla ponowienia komunikatu kanału.

Wyjście ponowienia komunikatu jest wyjściem wywoływanym przez MCA, gdy agent MCA odbierze kod zakończenia MQCC_FAILED z wywołania MQOPEN lub MQPUT. Celem wyjścia jest określenie przedziału czasu, przez który agent MCA oczekuje przed ponowną próbą wykonania operacji MQOPEN lub MQPUT. Alternatywnie można ustawić wyjście w taki sposób, aby nie podejmował ponownej próby wykonania operacji.

Wyjście jest wywoływane dla wszystkich kodów przyczyny, które mają kod zakończenia MQCC_FAILED- ustawienia wyjścia określają, które kody przyczyny agent MCA ma spróbować ponownie, dla ilu prób i w jakich odstępach czasu.

Jeśli operacja nie ma być więcej podejmowana, agent MCA wykonuje normalne przetwarzanie niepowodzenia. Przetwarzanie to obejmuje generowanie komunikatu raportu o wyjątku (jeśli został określony przez nadawcę) i umieszczanie oryginalnego komunikatu w kolejce niedostarczonych komunikatów lub usuwanie komunikatu (w zależności od tego, czy nadawca określił komunikat MQRO_DEAD_LETTER_Q, czy MQRO_DISCARD_MSG). Niepowodzenia związane z kolejką niedostarczonych komunikatów (na przykład pełna kolejka niedostarczonych komunikatów) nie powodują wywołania wyjścia ponowienia komunikatu.

Jeśli nazwa wyjścia nie jest pusta, wyjście jest wywoływane w następujących momentach:

- Bezpośrednio przed wykonaniem oczekiwania przed ponowną próbą dostarczenia komunikatu
- Podczas inicjowania i kończenia kanału

Opis zawartości tego pola w różnych środowiskach zawiera sekcja "MQCD-definicja kanału" na stronie 1526 .

To pole ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

Długość tego pola jest określona przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

Odstęp czasu MsgRetry(MQLONG)
To pole określa minimalny odstęp czasu (w milisekundach), po upływie którego podejmowana jest ponowna próba wykonania operacji otwierania lub umieszczania.

Efekt działania tego atrybutu zależy od tego, czy atrybut *MsgRetryExit* jest pusty, czy nie:

- Jeśli atrybut *MsgRetryExit* ma wartość pustą, atrybut **MsgRetryInterval** określa minimalny okres, przez jaki agent MCA czeka przed ponowieniem komunikatu, jeśli pierwsza operacja MQOPEN lub MQPUT nie powiedzie się i zostanie zwrócony kod zakończenia MQCC_FAILED. Wartość zero oznacza, że ponowna próba zostanie wykonana jak najszybciej po poprzedniej próbie. Ponowienia są wykonywane tylko wtedy, gdy wartość *MsgRetryCount* jest większa od zera.

  Ten atrybut jest również używany jako czas oczekiwania, jeśli wyjście ponowienia komunikatu zwraca niepoprawną wartość w polu *MsgRetryInterval* w MQCXP.
- Jeśli parametr *MsgRetryExit* nie jest pusty, atrybut **MsgRetryInterval** nie ma wpływu na agent MCA; zamiast tego jest wyjściem ponawiania komunikatu określającym czas oczekiwania agenta MCA. Atrybut **MsgRetryInterval** jest dostępny dla wyjścia w strukturze MQCD, ale wyjście nie musi go uznawać.

Wartość należy do zakresu od 0 do 999 999 999.

To pole ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

Następujące pola w tej strukturze nie są obecne, jeśli *Version* jest mniejsze niż MQCD_VERSION_4.

MsgRetryUserData (MQCHAR32)
To pole określa dane użytkownika wyjścia dla ponowienia komunikatu kanału.

Te dane są przekazywane do wyjścia ponowień komunikatu kanału w polu *ExitData* parametru **ChannelExitParms** (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji agenta MCA wszelkie zmiany wprowadzone w zawartości tego pola przez wyjście dowolnego typu są zachowywane przez agent MCA i widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Takie zmiany nie mają wpływu na definicję kanału używaną przez inne instancje MCA. Można użyć dowolnych znaków (w tym danych binarnych).

To pole ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR.

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_3.

To pole nie ma zastosowania w produkcie IBM MQ for IBM i.

MsgUserDane (MQCHAR32)
To pole określa dane użytkownika wyjścia komunikatów kanału.

Te dane są przekazywane do wyjścia komunikatu kanału w polu *ExitData* parametru **ChannelExitParms** (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji agenta MCA wszelkie zmiany wprowadzone w zawartości tego pola przez wyjście dowolnego typu są zachowywane przez agent MCA i widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Takie zmiany nie mają wpływu na definicję kanału używaną przez inne instancje MCA. Można użyć dowolnych znaków (w tym danych binarnych).

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma zastosowania w produkcie IBM MQ for IBM i.

MsgUserDataPtr (MQPTR)
To pole określa adres pierwszego pola *MsgUserData* .

Jeśli wartość *MsgExitsDefined* jest większa od zera, ten adres jest adresem listy elementów danych użytkownika dla każdego wyjścia komunikatu kanału w łańcuchu.

Każdy element danych użytkownika znajduje się w polu o długości *ExitDataLength*, dopełnianym do prawej strony odstępami. Istnieją pola *MsgExitsDefined* sąsiadujące ze sobą-po jednym dla każdego wyjścia. Jeśli liczba zdefiniowanych elementów danych użytkownika jest mniejsza niż liczba nazw wyjść, niezdefiniowane elementy danych użytkownika są puste. I odwrotnie, jeśli liczba zdefiniowanych elementów danych użytkownika jest większa niż liczba nazw wyjść, nadmiarowe elementy danych użytkownika są ignorowane i nie są prezentowane do wyjścia.

Wszystkie zmiany wprowadzone w tych wartościach przez wyjście są zachowywane. Pozwala to jednemu wyjściu przekazywać informacje do innego wyjścia. Sprawdzanie poprawności nie jest przeprowadzane dla żadnych zmian, dlatego na przykład dane binarne mogą być zapisywane w tych polach, jeśli jest to wymagane.

Jeśli *MsgExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

NetworkPriority (MQLONG)
Pole to określa priorytet połączenia sieciowego dla kanału.

Jeśli dostępnych jest wiele ścieżek do określonego miejsca docelowego, wybierana jest ścieżka o najwyższym priorytecie. Wartość należy do zakresu od 0 do 9; 0 oznacza najniższy priorytet.

To pole dotyczy tylko kanałów z *ChannelType* typu MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

Następujące pola w tej strukturze nie są dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

NonPersistentMsgSpeed (MQLONG)
To pole określa szybkość przesyłania nietrwałych komunikatów przez kanał.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER, MQCHT_REQUESTER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Jest to jedna z następujących wartości:

MQNPMS_NORMAL
Normalna prędkość.

Jeśli kanał jest zdefiniowany jako MQNPMS_NORMAL, komunikaty nietrwałe są przesyłane przez kanał z normalną szybkością. Ma to tę zaletę, że te komunikaty nie są tracone w przypadku awarii kanału. Ponadto komunikaty trwałe i nietrwałe w tej samej kolejce transmisji zachowują swoją kolejność względem siebie.

MQNPMS_FAST
Szybka szybkość.

Jeśli kanał jest zdefiniowany jako MQNPMS_FAST, komunikaty nietrwałe są przesyłane przez kanał z dużą szybkością. Zwiększa to przepustowość kanału, ale oznacza, że nietrwałe komunikaty są tracone w przypadku awarii kanału. Możliwe jest również, że komunikaty nietrwałe będą przeskakiwać przed trwałymi komunikatami oczekującymi w tej samej kolejce transmisji, co oznacza, że kolejność nietrwałych komunikatów nie będzie zachowana względem trwałych komunikatów. Zachowywana jest jednak kolejność komunikatów nietrwałych względem siebie. Podobnie zachowywana jest kolejność komunikatów trwałych względem siebie.

Hasło (MQCHAR12)
To pole określa hasło używane przez agenta kanału komunikatów podczas próby zainicjowania bezpiecznej sesji SNA ze zdalnym agentem kanału komunikatów.

To pole może być niepuste tylko w systemie AIX, Linux, and Windowsi ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER lub MQCHT_CLNTCONN. W systemie z/OSto pole nie jest istotne.

Długość tego pola jest określona przez wartość MQ_PASSWORD_LENGTH. Używane są jednak tylko pierwsze 10 znaków.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

PropertyControl (MQLONG)
To pole określa, co dzieje się z właściwościami komunikatów, gdy komunikat ma zostać wysłany do menedżera kolejek w wersji V6 lub wcześniejszej (menedżera kolejek, który nie rozumie pojęcia deskryptora właściwości).

Możliwe wartości:

MQPROP_KOMPATYBILNOŚĆ
Jeśli komunikat zawiera właściwość z przedrostkiem **mcd.**, **jms.**, **usr.** lub **mqext.**, wszystkie właściwości komunikatu są dostarczane do aplikacji w nagłówku MQRFH2 . W przeciwnym razie wszystkie właściwości komunikatu, z wyjątkiem tych, które są zawarte w deskryptorze komunikatu (lub rozszerzeniu), są usuwane i nie są już dostępne dla aplikacji.

Ta wartość jest wartością domyślną. Umożliwia ona aplikacjom, które oczekują, że właściwości powiązane z produktem JMSbędą znajdować się w nagłówku MQRFH2 w danych komunikatu, kontynuowanie pracy bez modyfikacji.

MQPROP_NONE
Wszystkie właściwości komunikatu, z wyjątkiem tych, które znajdują się w deskryptorze komunikatu (lub rozszerzeniu), są usuwane z komunikatu przed wysłaniem komunikatu do zdalnego menedżera kolejek.

MQPROP_ALL
Wszystkie właściwości komunikatu są dołączane do komunikatu podczas jego wysyłania do zdalnego menedżera kolejek. Właściwości te, z wyjątkiem tych, które znajdują się w deskryptorze komunikatu

(lub rozszerzeniu), zostają umieszczone w jednym lub większej liczbie nagłówków MQRFH2 danych komunikatu.

Ten atrybut ma zastosowanie do kanałów nadajnika, serwera, nadajnika klastra i odbiornika klastra.

“MQIA_ * (Selektory Atrybutów Całkowitoliczbowych)” na stronie 130
“MQPROP_ * (wartości kontrolne właściwości kolejki i kanału oraz maksymalna długość właściwości)” na stronie 170

PutAuthority (MQLONG)
To pole określa, czy identyfikator użytkownika w informacjach kontekstowych powiązanych z komunikatem jest używany do ustanowienia uprawnień do umieszczenia komunikatu w kolejce docelowej.

To pole ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_REQUESTER, MQCHT_RECEIVER lub MQCHT_CLUSRCVR. Jest to jedna z poniższych nazw:

MQPA_DEFAULT
Używany jest domyślny identyfikator użytkownika.

MQPA_CONTEXT
Używany jest identyfikator użytkownika kontekstu.

MQPA_ALTERNATE_OR_MCA
Używany jest identyfikator użytkownika z pola UserIdentifier deskryptora komunikatu. ID użytkownika odebrany z sieci nie jest używany. Ta wartość jest obsługiwana tylko w systemie z/OS.

MQPA_ONLY_MCA
Używany jest domyślny ID użytkownika. ID użytkownika odebrany z sieci nie jest używany. Ta wartość jest obsługiwana tylko w systemie z/OS.

QMgrName (MQCHAR48)
To pole określa nazwę menedżera kolejek, z którym może nawiązać połączenie wyjście.

W przypadku kanałów z atrybutem *ChannelType* innym niż MQCHT_CLNTCONN to pole jest nazwą menedżera kolejek, z którym może zostać nawiązane połączenie wyjście, a który w systemie AIX, Linux, and Windowsma zawsze wartość niepustą.

Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

ReceiveExit (MQCHARn)
To pole określa nazwę wyjścia odbierania kanału.

Jeśli ta nazwa nie jest pusta, wyjście jest wywoływane w następujących momentach:

- Bezpośrednio przed przetworzeniem odebranych danych sieciowych.

  Wyjście otrzymuje pełny bufor transmisji w postaci, w jakiej zostało odebrane. Zawartość buforu można modyfikować zgodnie z potrzebami.
- Podczas inicjowania i kończenia kanału.

Opis zawartości tego pola w różnych środowiskach zawiera sekcja “MQCD-definicja kanału” na stronie 1526 .

Długość tego pola jest określona przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

ReceiveExitPtr (MQPTR)
To pole określa adres pierwszego pola *ReceiveExit* .

Jeśli wartość parametru *ReceiveExitsDefined* jest większa od zera, adres ten jest adresem listy nazw wszystkich wyjść odbierania kanału w łańcuchu.

Każda nazwa znajduje się w polu o długości *ExitNameLength*, dopełnianym do prawej strony odstępami. Istnieją pola *ReceiveExitsDefined* sąsiadujące ze sobą-po jednym dla każdego wyjścia.

Wszystkie zmiany wprowadzone w tych nazwach przez wyjście są zachowywane, chociaż wyjście kanału komunikatów nie podejmuje jawnego działania-nie zmienia tego, które wyjścia są wywoływane.

Jeśli *ReceiveExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

ReceiveExitsZdefiniowane (MQLONG)
To pole określa liczbę wyjść odbierania kanału zdefiniowanych w łańcuchu.

Jest ona większa lub równa zero.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Dane ReceiveUser(MQCHAR32)
Ten kanał określa dane użytkownika wyjścia odbierania kanału.

Te dane są przekazywane do wyjścia odbierania kanału w polu *ExitData* parametru **ChannelExitParms** (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji agenta MCA wszelkie zmiany wprowadzone w zawartości tego pola przez wyjście dowolnego typu są zachowywane przez agent MCA i widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Dotyczy to wyjść z różnych konwersacji. Takie zmiany nie mają wpływu na definicję kanału używaną przez inne instancje MCA. Można użyć dowolnych znaków (w tym danych binarnych).

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma zastosowania w produkcie IBM MQ for IBM i.

Następujące pola w tej strukturze nie są dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

ReceiveUserDataPtr (MQPTR)
To pole określa adres pierwszego pola *ReceiveUserData* .

Jeśli wartość *ReceiveExitsDefined* jest większa od zera, adres ten jest adresem listy elementów danych użytkownika dla każdego wyjścia odbierania kanału w łańcuchu.

Każdy element danych użytkownika znajduje się w polu o długości *ExitDataLength*, dopełnianym do prawej strony odstępami. Istnieją pola *ReceiveExitsDefined* sąsiadujące ze sobą-po jednym dla każdego wyjścia. Jeśli liczba zdefiniowanych elementów danych użytkownika jest mniejsza niż liczba nazw wyjść, niezdefiniowane elementy danych użytkownika są puste. I odwrotnie, jeśli liczba zdefiniowanych elementów danych użytkownika jest większa niż liczba nazw wyjść, nadmiarowe elementy danych użytkownika są ignorowane i nie są prezentowane do wyjścia.

Wszystkie zmiany wprowadzone w tych wartościach przez wyjście są zachowywane. Pozwala to jednemu wyjściu przekazywać informacje do innego wyjścia. Sprawdzanie poprawności nie jest przeprowadzane dla żadnych zmian, dlatego na przykład dane binarne mogą być zapisywane w tych polach, jeśli jest to wymagane.

Jeśli *ReceiveExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Następujące pola w tej strukturze nie są dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_5.

RemotePassword (MQCHAR12)
To pole określa hasło od partnera.

To pole zawiera poprawne informacje tylko wtedy, gdy *ChannelType* to MQCHT_CLNTCONN lub MQCHT_SVRCONN.

- W przypadku wyjścia zabezpieczeń w kanale MQCHT_CLNTCONN to hasło jest hasłem uzyskanym ze środowiska. Wyjście może wysłać je do wyjścia zabezpieczeń na serwerze.
- W przypadku wyjścia zabezpieczeń w kanale MQCHT_SVRCONN to pole może zawierać hasło, które zostało uzyskane ze środowiska klienta, jeśli nie ma wyjścia zabezpieczeń klienta. Program zewnętrzny może użyć tego hasła do sprawdzenia poprawności identyfikatora użytkownika w programie *RemoteUserIdentifier*.

  Jeśli na kliencie istnieje wyjście zabezpieczeń, te informacje można uzyskać w przepływie zabezpieczeń od klienta.

Długość tego pola jest określona przez wartość MQ_PASSWORD_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

RemoteSecurityId (MQBYTE40)
Pole to określa identyfikator ochrony dla użytkownika zdalnego.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_CLNTCONN lub MQCHT_SVRCONN.

Następująca wartość specjalna wskazuje, że nie ma identyfikatora zabezpieczeń:

MQSID_BRAK
Nie określono identyfikatora zabezpieczeń.

Wartością długości pola jest zero binarne.

W przypadku języka programowania C stała MQSID_NONE_ARRAY jest również zdefiniowana. Ta stała ma taką samą wartość jak zmienna MQSID_NONE, ale jest tablicą znaków, a nie łańcuchem.

Jest to pole wejściowe do wyjścia. Długość tego pola jest określona przez wartość MQ_SECURITY_ID_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_6.

Poniższe pola w tej strukturze nie są dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

Identyfikator RemoteUser(MQCHAR12)
To pole określa pierwsze 12 bajtów identyfikatora użytkownika od partnera.

Istnieją dwa pola, które zawierają identyfikator użytkownika zdalnego:

- *RemoteUserIdentifier* zawiera pierwsze 12 bajtów identyfikatora użytkownika zdalnego i jest dopełniany odstępami, jeśli identyfikator jest krótszy niż 12 bajtów. Pole *RemoteUserIdentifier* może być puste.
- *LongRemoteUserIdPtr* wskazuje pełny identyfikator użytkownika zdalnego, który może być dłuższy niż 12 bajtów. Jego długość jest określona przez parametr *LongRemoteUserIdLength*. Pełny identyfikator nie zawiera końcowych odstępów i nie jest zakończony znakiem o kodzie zero. Jeśli identyfikator jest pusty, wartość *LongRemoteUserIdLength* wynosi zero, a wartość *LongRemoteUserIdPtr* jest niezdefiniowana.

  Parametr *LongRemoteUserIdPtr* nie jest obecny, jeśli wartość parametru *Version* jest mniejsza niż MQCD_VERSION_6.

Identyfikator użytkownika zdalnego jest odpowiedni tylko dla kanałów, których parametr *ChannelType* ma wartość MQCHT_CLNTCONN lub MQCHT_SVRCONN.

- W przypadku wyjścia zabezpieczeń w kanale MQCHT_CLNTCONN ta wartość jest identyfikatorem użytkownika uzyskanym ze środowiska. Wyjście może wysłać je do wyjścia zabezpieczeń na serwerze.

- W przypadku wyjścia zabezpieczeń w kanale MQCHT_SVRCONN pole to może zawierać identyfikator użytkownika uzyskany ze środowiska klienta, jeśli nie istnieje wyjście zabezpieczeń klienta. Program zewnętrzny może sprawdzić poprawność tego identyfikatora użytkownika (prawdopodobnie z hasłem w pliku *RemotePassword*) i zaktualizować wartość w pliku *MCAUserIdentifier*.

  Jeśli na kliencie istnieje wyjście zabezpieczeń, te informacje można uzyskać w przepływie zabezpieczeń od klienta.

Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

SecurityExit (MQCHARn)
To pole określa nazwę wyjścia zabezpieczeń kanału.

Jeśli ta nazwa nie jest pusta, wyjście jest wywoływane w następujących momentach:

- Natychmiast po uruchomieniu kanału.

  Przed przesłaniem jakiegokolwiek komunikatu wyjście ma możliwość sprawdzenia przepływów zabezpieczeń w celu sprawdzenia poprawności autoryzacji połączenia.
- Po odebraniu odpowiedzi na przepływ komunikatów zabezpieczeń.

  Wszystkie przepływy komunikatów zabezpieczeń odebrane ze zdalnego procesora na komputerze zdalnym są przesyłane do wyjścia.
- Podczas inicjowania i kończenia kanału.

Opis zawartości tego pola w różnych środowiskach zawiera sekcja "MQCD-definicja kanału" na stronie 1526 .

Długość tego pola jest określona przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

Dane SecurityUser(MQCHAR32)
Ten kanał określa dane użytkownika wyjścia zabezpieczeń kanału.

Te dane są przekazywane do wyjścia zabezpieczeń kanału w polu *ExitData* parametru **ChannelExitParms** (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji agenta MCA wszelkie zmiany wprowadzone w zawartości tego pola przez wyjście dowolnego typu są zachowywane przez agent MCA i widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Dotyczy to wyjść z różnych konwersacji. Takie zmiany nie mają wpływu na definicję kanału używaną przez inne instancje MCA. Można użyć dowolnych znaków (w tym danych binarnych).

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma zastosowania w produkcie IBM MQ for IBM i.

SendExit (MQCHARn)
To pole określa nazwę wyjścia wysyłania kanału.

Jeśli ta nazwa nie jest pusta, wyjście jest wywoływane w następujących momentach:

- Bezpośrednio przed wysłaniem danych do sieci.

  Wyjście otrzymuje pełny bufor transmisji przed jego przesłaniem. Zawartość buforu można modyfikować zgodnie z potrzebami.
- Podczas inicjowania i kończenia kanału.

Opis zawartości tego pola w różnych środowiskach zawiera sekcja "MQCD-definicja kanału" na stronie 1526 .

Długość tego pola jest określona przez wartość MQ_EXIT_NAME_LENGTH.

Uwaga: Wartość tej stałej jest specyficzna dla środowiska.

SendExitPtr (MQPTR)
To pole określa adres pierwszego pola *SendExit* .

Jeśli wartość *SendExitsDefined* jest większa od zera, ten adres jest adresem listy nazw poszczególnych wyjść wysyłania kanału w łańcuchu.

Każda nazwa znajduje się w polu o długości *ExitNameLength*, dopełnianym do prawej strony odstępami. Istnieją pola *SendExitsDefined* sąsiadujące ze sobą-po jednym dla każdego wyjścia.

Wszystkie zmiany wprowadzone w tych nazwach przez wyjście są zachowywane, chociaż wyjście wysyłania komunikatu nie wykonuje jawnego działania-nie zmienia wywoływanych wyjść.

Jeśli *SendExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

SendExitsZdefiniowane (MQLONG)
To pole określa liczbę wyjść wysyłania kanału zdefiniowanych w łańcuchu.

Jest ona większa lub równa zero.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

Dane SendUser(MQCHAR32)
To pole określa dane użytkownika wyjścia wysyłania kanału.

Te dane są przekazywane do wyjścia wysyłania kanału w polu *ExitData* parametru **ChannelExitParms** (patrz MQ_CHANNEL_EXIT).

To pole początkowo zawiera dane, które zostały ustawione w definicji kanału. Jednak w czasie życia tej instancji agenta MCA wszelkie zmiany wprowadzone w zawartości tego pola przez wyjście dowolnego typu są zachowywane przez agent MCA i widoczne dla kolejnych wywołań wyjść (niezależnie od typu) dla tej instancji agenta MCA. Dotyczy to wyjść z różnych konwersacji. Takie zmiany nie mają wpływu na definicję kanału używaną przez inne instancje MCA. Można użyć dowolnych znaków (w tym danych binarnych).

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH.

To pole nie ma zastosowania w produkcie IBM MQ for IBM i.

SendUserDataPtr (MQPTR)
To pole określa adres pola *SendUserData* .

Jeśli wartość *SendExitsDefined* jest większa od zera, ten adres jest adresem listy elementów danych użytkownika dla każdego wyjścia komunikatu kanału w łańcuchu.

Każdy element danych użytkownika znajduje się w polu o długości *ExitDataLength*, dopełnianym do prawej strony odstępami. Istnieją pola *MsgExitsDefined* sąsiadujące ze sobą-po jednym dla każdego wyjścia. Jeśli liczba zdefiniowanych elementów danych użytkownika jest mniejsza niż liczba nazw wyjść, niezdefiniowane elementy danych użytkownika są puste. I odwrotnie, jeśli liczba zdefiniowanych elementów danych użytkownika jest większa niż liczba nazw wyjść, nadmiarowe elementy danych użytkownika są ignorowane i nie są prezentowane do wyjścia.

Wszystkie zmiany wprowadzone w tych wartościach przez wyjście są zachowywane. Pozwala to jednemu wyjściu przekazywać informacje do innego wyjścia. Sprawdzanie poprawności nie jest przeprowadzane dla żadnych zmian, dlatego na przykład dane binarne mogą być zapisywane w tych polach, jeśli jest to wymagane.

Jeśli *SendExitsDefined* ma wartość zero, to pole jest wskaźnikiem pustym.

Na platformach, na których język programowania nie obsługuje typu danych wskaźnika, pole to jest zadeklarowane jako łańcuch bajtowy o odpowiedniej długości.

Jest to pole wejściowe do wyjścia. To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

SeqNumberOpływanie (MQLONG)
To pole określa najwyższy dozwolony numer kolejny komunikatu.

Po osiągnięciu tej wartości numery kolejne są zawijane w celu ponownego rozpoczęcia od 1.

Ta wartość nie podlega negocjacjom i musi być zgodna zarówno w definicji kanału lokalnego, jak i zdalnego.

To pole nie dotyczy kanałów z wartością *ChannelType* wynoszącą MQCHT_SVRCONN lub MQCHT_CLNTCONN.

SharingConversations (MQLONG)
To pole określa maksymalną liczbę konwersacji, które mogą współużytkować instancję kanału powiązaną z tym kanałem.

To pole jest używane w kanałach połączenia klienta i połączenia serwera.

Wartość 0 oznacza, że kanał działa tak jak w wersjach wcześniejszych niż IBM WebSphere MQ 7.0 w odniesieniu do następujących atrybutów:

- Współużytkowanie konwersacji
- Odczyt z wyprzedzeniem
- `STOP CHANNEL(`*channelname*`) MODE(QUIESCE)`
- Pulsowanie
- Wykorzystanie asynchroniczne klienta

Wartość 1 jest minimalną wartością zachowania IBM MQ . Chociaż w instancji kanału dozwolona jest tylko jedna konwersacja, dostępne są operacje odczytu z wyprzedzeniem, wykorzystania asynchronicznego oraz zachowanie funkcji pulsu programu `CLNTCONN-SVRCONN` i zatrzymywania kanału wyciszonego.

Jest to pole wejściowe do wyjścia. Nie występuje, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_9.

Wartością domyślną tego pola jest 10.

Uwaga: Limity *MaxInstances* i *MaxInstancesPerClient* zastosowane dla kanału ograniczają liczbę instancji kanału, a nie liczbę konwersacji, które mogą współużytkować te instancje.

ShortConnectionNazwa (MQCHAR20)
Pole to określa pierwsze 20 bajtów nazwy połączenia.

Jeśli pole *Version* ma wartość MQCD_VERSION_1, *ShortConnectionName* zawiera pełną nazwę połączenia.

Jeśli pole *Version* ma wartość MQCD_VERSION_2 lub większą, *ShortConnectionName* zawiera pierwsze 20 znaków nazwy połączenia. Pełna nazwa połączenia jest podana w polu *ConnectionName* ; *ShortConnectionName* i pierwsze 20 znaków *ConnectionName* są identyczne.

Szczegółowe informacje na temat zawartości tego pola zawiera sekcja *ConnectionName* .

Uwaga: Nazwa tego pola została zmieniona dla MQCD_VERSION_2 i kolejnych wersji MQCD; pole to miało wcześniej nazwę *ConnectionName*.

Długość tego pola jest określona przez wartość MQ_SHORT_CONN_NAME_LENGTH.

ShortRetry(MQLONG)
To pole określa maksymalną liczbę prób nawiązania połączenia z komputerem zdalnym.

To pole określa maksymalną liczbę prób nawiązania połączenia z komputerem zdalnym, podejmowanych w odstępach czasu określonych przez parametr *ShortRetryInterval*, zanim zostaną użyte (zwykle dłuższe) *LongRetryCount* i *LongRetryInterval* .

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

Odstęp czasu ShortRetry(MQLONG)
To pole określa maksymalną liczbę sekund oczekiwania przed ponowną próbą nawiązania połączenia z komputerem zdalnym.

Odstęp czasu między ponownymi próbami może zostać przedłużony, jeśli kanał ma oczekiwać na aktywowanie.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_CLUSSDR lub MQCHT_CLUSRCVR.

z/OS *Zabezpieczenie SPLProtection (MQLONG)*
Pole to określa wartość ochrony strategii bezpieczeństwa systemu AMS .

Jest to jedna z następujących wartości:

MQSPL_PASSTHRU
Tranzyt, bez zmian, wszystkie komunikaty wysłane lub odebrane przez agent MCA dla tego kanału.

Ta wartość jest odpowiednia tylko dla kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER, MQCHT_SERVER, MQCHT_RECEIVER lub MQCHT_REQUESTER i jest wartością domyślną.

MQSPL_REMOVE
Usuń wszelkie zabezpieczenia AMS przed komunikatami pobranymi z kolejki transmisji przez agent MCA i wyślij komunikaty do partnera.

Ta wartość ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER lub MQCHT_SERVER.

MQSPL_ASPOLICY
W przypadku wybrania tej wartości względem komunikatów przychodzących będzie stosowana ochrona AMS określana na podstawie strategii zdefiniowanej dla kolejki docelowej przed umieszczeniem ich w kolejce docelowej.

Ta wartość ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_RECEIVER lub MQCHT_REQUESTER.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_12.

SSLCipherSpec (MQCHAR32)
To pole określa specyfikację szyfru, która jest używana podczas używania protokołu TLS.

Jeśli parametr SSLCipherSpec jest pusty, kanał nie używa protokołu TLS. Jeśli pole nie jest puste, zawiera łańcuch określający używany parametr CipherSpec .

Ten parametr jest poprawny dla wszystkich typów kanałów. Jest on obsługiwany na następujących platformach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows
- z/OS z/OS

Jest ona poprawna tylko dla typów kanałów typu transportu (TRPTYPE) TCP.

Jest to pole wejściowe do wyjścia. Długość tego pola jest określona przez wartość MQ_SSL_CIPHER_SPEC_LENGTH. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

SSLClientAuth (MQLONG),
To pole określa, czy wymagane jest uwierzytelnianie klienta TLS.

To pole dotyczy tylko definicji kanału SVRCONN.

Jest to jedna z następujących wartości:

MQSCA_XX_ENCODE_CASE_ONE wymagane
Wymagane jest uwierzytelnienie klienta.

MQSCA_OPCJONALNA
Uwierzytelnianie klienta jest opcjonalne.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

SSLPeerNameDługość (MQLONG)
To pole określa długość (w bajtach) nazwy węzła TLS wskazywanej przez *SSLPeerNamePtr*.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

SSLPeerNamePtr (MQPTR)
To pole określa adres nazwy węzła TLS.

Gdy certyfikat zostanie odebrany podczas pomyślnego uzgadniania TLS, nazwa wyróżniająca podmiotu certyfikatu jest kopiowana do pola MQCD dostępnego za pośrednictwem atrybutu SSLPeerNamePtr na końcu kanału, który odbiera certyfikat. Nadpisuje ona wartość SSLPeerName dla kanału, jeśli ta wartość jest obecna w definicji kanału użytkownika lokalnego. Jeśli na tym końcu kanału określono wyjście zabezpieczeń, otrzymuje ono nazwę wyróżniającą z certyfikatu węzła sieci w MQCD.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_7.

Uwaga: Aplikacje wyjścia zabezpieczeń utworzone przed wydaniem produktu IBM WebSphere MQ 7.1 mogą wymagać aktualizacji. Więcej informacji na ten temat zawiera sekcja Programy obsługi wyjścia zabezpieczeń kanału.

StrucLength (MQLONG)
To pole określa długość (w bajtach) struktury MQCD.

Długość nie obejmuje żadnego z łańcuchów adresowanych przez pola wskaźnika znajdujące się w strukturze. Jest to jedna z następujących wartości:

MQCD_LENGTH_4
Długość struktury definicji kanału version-4 .

MQCD_LENGTH_5
Długość struktury definicji kanału version-5 .

MQCD_LENGTH_6
Długość struktury definicji kanału version-6 .

MQCD_LENGTH_7
Długość struktury definicji kanału version-7 .

MQCD_LENGTH_8
Długość struktury definicji kanału version-8 .

MQCD_LENGTH_9
Długość struktury definicji kanału version-9 .

MQCD_LENGTH_10
Długość struktury definicji kanału version-10 .

MQCD_LENGTH_11
Długość struktury definicji kanału version-11 .

z/OS **MQCD_LENGTH_12**
Długość struktury definicji kanału version-12 .

Następująca stała określa długość bieżącej wersji:

MQCD_CURRENT_LENGTH
Długość bieżącej wersji struktury definicji kanału.

Uwaga: Stałe te mają wartości specyficzne dla środowiska.

To pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_4.

TpName (MQCHAR64)
Pole to określa nazwę programu transakcyjnego LU 6.2 .

To pole ma zastosowanie tylko wtedy, gdy protokołem transmisji (*TransportType*) jest MQXPT_LU62, a *ChannelType* nie jest MQCHT_SVRCONN ani MQCHT_RECEIVER.

To pole jest zawsze puste na platformach, na których informacje są zawarte w obiekcie pobocznym komunikacji.

Długość tego pola jest określona przez wartość MQ_TP_NAME_LENGTH.

TransportType (MQLONG)
To pole określa protokół transmisji, który ma być używany.

Wartość nie jest sprawdzana, jeśli kanał został zainicjowany z drugiego końca.

Jest to jedna z następujących wartości:

MQXPT_LU62
Protokół transportowy LU 6.2 .

MQXPT_TCP
Protokół transportowy TCP/IP.

MQXPT_NETBIOS
Protokół transportowy NetBIOS .

Ta wartość jest obsługiwana w następujących środowiskach: Windows.

MQXPT_SPX
Protokół transportowy SPX.

Ta wartość jest obsługiwana w następujących środowiskach: Windowsoraz klienty IBM MQ połączone z tymi systemami.

UseDLQ (MQLONG)
To pole określa, czy używana jest kolejka niedostarczonych komunikatów (lub kolejka niedostarczonych komunikatów), gdy komunikaty nie mogą być dostarczane przez kanały.

Może on zawierać jedną z następujących wartości:

MQUSEDLQ_NO
Komunikaty, które nie mogą być dostarczane przez kanał, są traktowane jako niepowodzenie. Kanał usuwa komunikat lub kończy kanał zgodnie z ustawieniem NPMSPEED.

MQUSEDLQ_TAK
Jeśli atrybut menedżera kolejek DEADQ udostępnia nazwę kolejki niedostarczonych komunikatów, to jest ona używana, w przeciwnym razie zachowanie jest takie samo jak w przypadku atrybutu NO. Wartością domyślną jest YES.

UserIdentifier (MQCHAR12)
To pole określa identyfikator użytkownika używany przez agenta kanału komunikatów podczas próby zainicjowania bezpiecznej sesji SNA ze zdalnym agentem kanału komunikatów.

To pole może być niepuste tylko w systemie AIX, Linux, and Windowsi ma zastosowanie tylko w przypadku kanałów z atrybutem *ChannelType* o wartości MQCHT_SENDER, MQCHT_SERVER, MQCHT_REQUESTER lub MQCHT_CLNTCONN. W systemie z/OSto pole nie jest istotne.

Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH. Używane są jednak tylko pierwsze 10 znaków.

To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCD_VERSION_2.

Wersja (MQLONG)
Pole Version określa najwyższy numer wersji, który można ustawić dla struktury.

Wartość zależy od środowiska:

MQCD _VERSION_1
Struktura definicji kanału w wersji 1.

MQCD _VERSION_2
Struktura definicji kanału w wersji 2.

MQCD _VERSION_3
Struktura definicji kanału w wersji 3.

MQCD _VERSION_4
Struktura definicji kanału wersji 4.

MQCD _VERSION_5
Struktura definicji kanału w wersji 5.

MQCD _VERSION_6
Struktura definicji kanału w wersji 6.

MQCD _VERSION_7
Struktura definicji kanału w wersji 7.

MQCD _VERSION_8
Struktura definicji kanału w wersji 8.

MQCD _VERSION_9
Struktura definicji kanału w wersji 9.

MQCD _VERSION_10
Struktura definicji kanału w wersji 10.

MQCD _VERSION_11
Struktura definicji kanału w wersji 11.

Wersja 11 jest najwyższą wartością, jaką można ustawić w tym polu w systemie IBM MQ 8.0 na wszystkich platformach.

z/OS **MQCD _VERSION_12**
Struktura definicji kanału w wersji 12.

Wersja 12 jest najwyższą wartością, jaką można ustawić w tym polu w systemie IBM MQ 9.1.3.

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQCD_CURRENT_VERSION
Wartość ustawiona w pliku MQCD_CURRENT_VERSION jest bieżącą wersją używanej struktury definicji kanału.

Wartość parametru MQCD_CURRENT_VERSION zależy od środowiska. Zawiera najwyższą wartość obsługiwaną przez platformę.

Parametr MQCD_CURRENT_VERSION nie jest używany do inicjowania struktur domyślnych udostępnionych w nagłówku, kopii i plikach włączanych w różnych językach programowania. Domyślne inicjowanie Version zależy od platformy i wersji.

Deklaracje MQCD w nagłówkach, kopiach i plikach włączanych są inicjowane jako MQCD_VERSION_6. Aby użyć dodatkowych pól MQCD , aplikacje muszą ustawić numer wersji na wartość MQCD_CURRENT_VERSION. W przypadku tworzenia aplikacji, która jest przenośna między kilkoma środowiskami, należy wybrać wersję, która jest obsługiwana we wszystkich środowiskach.

Wskazówka: Po wprowadzeniu nowej wersji struktury MQCD układ istniejącej części nie ulega zmianie. Wyjście musi sprawdzić numer wersji. Musi być równa lub większa od najniższej wersji zawierającej pola, których ma używać wyjście.

XmitQName (MQCHAR48)
Pole to określa nazwę kolejki transmisji, z której pobierane są komunikaty.

To pole ma zastosowanie tylko w przypadku kanałów, których parametr *ChannelType* ma wartość MQCHT_SENDER lub MQCHT_SERVER.

Długość tego pola jest określona przez wartość MQ_Q_NAME_LENGTH.

Deklaracja C

Ta deklaracja jest deklaracją C dla struktury MQCD.

```
typedef struct tagMQCD MQCD;
typedef MQCD MQPOINTER PMQCD;
typedef PMQCD MQPOINTER PPMQCD;

 struct tagMQCD {
   MQCHAR    ChannelName[20];           /* Channel definition name */
   MQLONG    Version;                   /* Structure version number */
   MQLONG    ChannelType;               /* Channel type */
   MQLONG    TransportType;             /* Transport type */
   MQCHAR    Desc[64];                  /* Channel description */
   MQCHAR    QMgrName[48];              /* Queue manager name */
   MQCHAR    XmitQName[48];             /* Transmission queue name */
   MQCHAR    ShortConnectionName[20];   /* First 20 bytes of */
                                        /* connection name */
   MQCHAR    MCAName[20];               /* Reserved */
   MQCHAR    ModeName[8];               /* LU 6.2 Mode name */
   MQCHAR    TpName[64];                /* LU 6.2 transaction program */
                                        /* name */
   MQLONG    BatchSize;                 /* Batch size */
   MQLONG    DiscInterval;              /* Disconnect interval */
   MQLONG    ShortRetryCount;           /* Short retry count */
   MQLONG    ShortRetryInterval;        /* Short retry wait interval */
   MQLONG    LongRetryCount;            /* Long retry count */
   MQLONG    LongRetryInterval;         /* Long retry wait interval */
   MQCHAR    SecurityExit[128];         /* Channel security exit name */
   MQCHAR    MsgExit[128];              /* Channel message exit name */
   MQCHAR    SendExit[128];             /* Channel send exit name */
   MQCHAR    ReceiveExit[128];          /* Channel receive exit name */
   MQLONG    SeqNumberWrap;             /* Highest allowable message */
                                        /* sequence number */
   MQLONG    MaxMsgLength;              /* Maximum message length */
   MQLONG    PutAuthority;              /* Put authority */
   MQLONG    DataConversion;            /* Data conversion */
   MQCHAR    SecurityUserData[32];      /* Channel security exit user */
                                        /* data */
   MQCHAR    MsgUserData[32];           /* Channel message exit user */
                                        /* data */
   MQCHAR    SendUserData[32];          /* Channel send exit user */
                                        /* data */
   MQCHAR    ReceiveUserData[32];       /* Channel receive exit user */
                                        /* data */
   /* Ver:1 */
   MQCHAR    UserIdentifier[12];        /* User identifier */
   MQCHAR    Password[12];              /* Password */
   MQCHAR    MCAUserIdentifier[12];     /* First 12 bytes of MCA user */
                                        /* identifier */
   MQLONG    MCAType;                   /* Message channel agent type */
   MQCHAR    ConnectionName[264];       /* Connection name */
   MQCHAR    RemoteUserIdentifier[12];  /* First 12 bytes of user */
                                        /* identifier from partner */
```

```
MQCHAR    RemotePassword[12];        /* Password from partner */
/* Ver:2 */
MQCHAR    MsgRetryExit[128];         /* Channel message retry exit */
                                     /* name */
MQCHAR    MsgRetryUserData[32];      /* Channel message retry exit */
                                     /* user data */
MQLONG    MsgRetryCount;             /* Number of times MCA will */
                                     /* try to put the message, */
                                     /* after first attempt has */
                                     /* failed */
MQLONG    MsgRetryInterval;          /* Minimum interval in */
                                     /* milliseconds after which */
                                     /* the open or put operation */
                                     /* will be retried */
/* Ver:3 */
MQLONG    HeartbeatInterval;         /* Time in seconds between */
                                     /* heartbeat flows */
MQLONG    BatchInterval;             /* Batch duration */
MQLONG    NonPersistentMsgSpeed;     /* Speed at which */
                                     /* nonpersistent messages are */
                                     /* sent */
MQLONG    StrucLength;               /* Length of MQCD structure */
MQLONG    ExitNameLength;            /* Length of exit name */
MQLONG    ExitDataLength;            /* Length of exit user data */
MQLONG    MsgExitsDefined;           /* Number of message exits */
                                     /* defined */
MQLONG    SendExitsDefined;          /* Number of send exits */
                                     /* defined */
MQLONG    ReceiveExitsDefined;       /* Number of receive exits */
                                     /* defined */
MQPTR     MsgExitPtr;                /* Address of first MsgExit */
                                     /* field */
MQPTR     MsgUserDataPtr;            /* Address of first */
                                     /* MsgUserData field */
MQPTR     SendExitPtr;               /* Address of first SendExit */
                                     /* field */
MQPTR     SendUserDataPtr;           /* Address of first */
                                     /* SendUserData field */
MQPTR     ReceiveExitPtr;            /* Address of first */
                                     /* ReceiveExit field */
MQPTR     ReceiveUserDataPtr;        /* Address of first */
                                     /* ReceiveUserData field */
/* Ver:4 */
MQPTR     ClusterPtr;                /* Address of a list of */
                                     /* cluster names */
MQLONG    ClustersDefined;           /* Number of clusters to */
                                     /* which the channel belongs */
MQLONG    NetworkPriority;           /* Network priority */
/* Ver:5 */
MQLONG    LongMCAUserIdLength;       /* Length of long MCA user */
                                     /* identifier */
MQLONG    LongRemoteUserIdLength;    /* Length of long remote user */
                                     /* identifier */
MQPTR     LongMCAUserIdPtr;          /* Address of long MCA user */
                                     /* identifier */
MQPTR     LongRemoteUserIdPtr;       /* Address of long remote */
                                     /* user identifier */
MQBYTE40  MCASecurityId;             /* MCA security identifier */
MQBYTE40  RemoteSecurityId;          /* Remote security identifier */
/* Ver:6 */
MQCHAR    SSLCipherSpec[32];         /* TLS CipherSpec */
MQPTR     SSLPeerNamePtr;            /* Address of TLS peer name */
MQLONG    SSLPeerNameLength;         /* Length of TLS peer name */
MQLONG    SSLClientAuth;             /* Whether TLS client */
                                     /* authentication is required */
MQLONG    KeepAliveInterval;         /* Keepalive interval */
MQCHAR    LocalAddress[48];          /* Local communications */
                                     /* address */
MQLONG    BatchHeartbeat;            /* Batch heartbeat interval */
/* Ver:7 */
MQLONG    HdrCompList[2];            /* Header data compression */
                                     /* list */
MQLONG    MsgCompList[16];           /* Message data compression */
                                     /* list */
MQLONG    CLWLChannelRank;           /* Channel rank */
MQLONG    CLWLChannelPriority;       /* Channel priority */
MQLONG    CLWLChannelWeight;         /* Channel weight */
MQLONG    ChannelMonitoring;         /* Channel monitoring */
MQLONG    ChannelStatistics;         /* Channel statistics */
/* Ver:8 */
MQLONG    SharingConversations;      /* Limit on sharing */
                                     /* conversations */
```

```
   MQLONG    PropertyControl;          /* Message property control */
   MQLONG    MaxInstances;             /* Limit on SVRCONN channel */
                                       /* instances */
   MQLONG    MaxInstancesPerClient;    /* Limit on SVRCONN channel */
                                       /* instances per client */
   MQLONG    ClientChannelWeight;      /* Client channel weight */
   MQLONG    ConnectionAffinity;       /* Connection affinity */
   /* Ver:9 */
   MQLONG    BatchDataLimit;           /* Batch data limit */
   MQLONG    UseDLQ;                   /* Use Dead Letter Queue */
   MQLONG    DefReconnect;             /* Default client reconnect */
                                       /* option */
   /* Ver:10 */
   MQCHAR64  CertificateLabel;         /* Certificate label */
 /* Ver:11 */
   MQLONG    SPLProtection             /* AMS Security policy protection */
 /* Ver:12 */
};
```

Deklaracja języka COBOL

Ta deklaracja jest deklaracją języka COBOL dla struktury MQCD.

```
** MQCD structure
       10  MQCD.
      ** Channel definition name
       15  MQCD-CHANNELNAME PIC X(20).
      ** Structure version number
       15  MQCD-VERSION PIC S9(9) BINARY.
      ** Channel type
       15  MQCD-CHANNELTYPE PIC S9(9) BINARY.
      ** Transport type
       15  MQCD-TRANSPORTTYPE PIC S9(9) BINARY.
      ** Channel description
       15  MQCD-DESC PIC X(64).
      ** Queue manager name
       15  MQCD-QMGRNAME PIC X(48).
      ** Transmission queue name
       15  MQCD-XMITQNAME PIC X(48).
      ** First 20 bytes of connection name
       15  MQCD-SHORTCONNECTIONNAME PIC X(20).
      ** Reserved
       15  MQCD-MCANAME PIC X(20).
      ** LU 6.2 Mode name
       15  MQCD-MODENAME PIC X(8).
      ** LU 6.2 transaction program name
       15  MQCD-TPNAME PIC X(64).
      ** Batch size
       15  MQCD-BATCHSIZE PIC S9(9) BINARY.
      ** Disconnect interval
       15  MQCD-DISCINTERVAL PIC S9(9) BINARY.
      ** Short retry count
       15  MQCD-SHORTRETRYCOUNT PIC S9(9) BINARY.
      ** Short retry wait interval
       15  MQCD-SHORTRETRYINTERVAL PIC S9(9) BINARY.
      ** Long retry count
       15  MQCD-LONGRETRYCOUNT PIC S9(9) BINARY.
      ** Long retry wait interval
       15  MQCD-LONGRETRYINTERVAL PIC S9(9) BINARY.
      ** Channel security exit name
       15  MQCD-SECURITYEXIT PIC X(20).
      ** Channel message exit name
       15  MQCD-MSGEXIT PIC X(20).
      ** Channel send exit name
       15  MQCD-SENDEXIT PIC X(20).
      ** Channel receive exit name
       15  MQCD-RECEIVEEXIT PIC X(20).
      ** Highest allowable message sequence number
       15  MQCD-SEQNUMBERWRAP PIC S9(9) BINARY.
      ** Maximum message length
       15  MQCD-MAXMSGLENGTH PIC S9(9) BINARY.
      ** Put authority
       15  MQCD-PUTAUTHORITY PIC S9(9) BINARY.
      ** Data conversion
       15  MQCD-DATACONVERSION PIC S9(9) BINARY.
      ** Channel security exit user data
       15  MQCD-SECURITYUSERDATA PIC X(32).
      ** Channel message exit user data
       15  MQCD-MSGUSERDATA PIC X(32).
```

```
      ** Channel send exit user data
       15  MQCD-SENDUSERDATA PIC X(32).
      ** Channel receive exit user data
       15  MQCD-RECEIVEUSERDATA PIC X(32).
      ** Ver:1 **
      ** User identifier
       15  MQCD-USERIDENTIFIER PIC X(12).
      ** Password
       15  MQCD-PASSWORD PIC X(12).
      ** First 12 bytes of MCA user identifier
       15  MQCD-MCAUSERIDENTIFIER PIC X(12).
      ** Message channel agent type
       15  MQCD-MCATYPE PIC S9(9) BINARY.
      ** Connection name
       15  MQCD-CONNECTIONNAME PIC X(264).
      ** First 12 bytes of user identifier from partner
       15  MQCD-REMOTEUSERIDENTIFIER PIC X(12).
      ** Password from partner
       15  MQCD-REMOTEPASSWORD PIC X(12).
      ** Ver:2 **
      ** Channel message retry exit name
       15  MQCD-MSGRETRYEXIT PIC X(20).
      ** Channel message retry exit user data
       15  MQCD-MSGRETRYUSERDATA PIC X(32).
      ** Number of times MCA will try to put the message, after first
      ** attempt has failed
       15  MQCD-MSGRETRYCOUNT PIC S9(9) BINARY.
      ** Minimum interval in milliseconds after which the open or put
      ** operation will be retried
       15  MQCD-MSGRETRYINTERVAL PIC S9(9) BINARY.
      ** Ver:3 **
      ** Time in seconds between heartbeat flows
       15  MQCD-HEARTBEATINTERVAL PIC S9(9) BINARY.
      ** Batch duration
       15  MQCD-BATCHINTERVAL PIC S9(9) BINARY.
      ** Speed at which nonpersistent messages are sent
       15  MQCD-NONPERSISTENTMSGSPEED PIC S9(9) BINARY.
      ** Length of MQCD structure
       15  MQCD-STRUCLENGTH PIC S9(9) BINARY.
      ** Length of exit name
       15  MQCD-EXITNAMELENGTH PIC S9(9) BINARY.
      ** Length of exit user data
       15  MQCD-EXITDATALENGTH PIC S9(9) BINARY.
      ** Number of message exits defined
       15  MQCD-MSGEXITSDEFINED PIC S9(9) BINARY.
      ** Number of send exits defined
       15  MQCD-SENDEXITSDEFINED PIC S9(9) BINARY.
      ** Number of receive exits defined
       15  MQCD-RECEIVEEXITSDEFINED PIC S9(9) BINARY.
      ** Address of first MsgExit field
       15  MQCD-MSGEXITPTR POINTER.
      ** Address of first MsgUserData field
       15  MQCD-MSGUSERDATAPTR POINTER.
      ** Address of first SendExit field
       15  MQCD-SENDEXITPTR POINTER.
      ** Address of first SendUserData field
       15  MQCD-SENDUSERDATAPTR POINTER.
      ** Address of first ReceiveExit field
       15  MQCD-RECEIVEEXITPTR POINTER.
      ** Address of first ReceiveUserData field
       15  MQCD-RECEIVEUSERDATAPTR POINTER.
      ** Ver:4 **
      ** Address of a list of cluster names
       15  MQCD-CLUSTERPTR POINTER.
      ** Number of clusters to which the channel belongs
       15  MQCD-CLUSTERSDEFINED PIC S9(9) BINARY.
      ** Network priority
       15  MQCD-NETWORKPRIORITY PIC S9(9) BINARY.
      ** Ver:5 **
      ** Length of long MCA user identifier
       15  MQCD-LONGMCAUSERIDLENGTH PIC S9(9) BINARY.
      ** Length of long remote user identifier
       15  MQCD-LONGREMOTEUSERIDLENGTH PIC S9(9) BINARY.
      ** Address of long MCA user identifier
       15  MQCD-LONGMCAUSERIDPTR POINTER.
      ** Address of long remote user identifier
       15  MQCD-LONGREMOTEUSERIDPTR POINTER.
      ** MCA security identifier
       15  MQCD-MCASECURITYID PIC X(40).
      ** Remote security identifier
       15  MQCD-REMOTESECURITYID PIC X(40).
      ** Ver:6 **
```

```
      ** TLS CipherSpec
       15  MQCD-SSLCIPHERSPEC PIC X(32).
      ** Address of TLS peer name
       15  MQCD-SSLPEERNAMEPTR POINTER.
      ** Length of TLS peer name
       15  MQCD-SSLPEERNAMELENGTH PIC S9(9) BINARY.
      ** Whether TLS client authentication is required
       15  MQCD-SSLCLIENTAUTH PIC S9(9) BINARY.
      ** Keepalive interval
       15  MQCD-KEEPALIVEINTERVAL PIC S9(9) BINARY.
      ** Local communications address
       15  MQCD-LOCALADDRESS PIC X(48).
      ** Batch heartbeat interval
       15  MQCD-BATCHHEARTBEAT PIC S9(9) BINARY.
      ** Ver:7 **
      ** Header data compression list
       15  MQCD-HDRCOMPLIST PIC S9(9) BINARY.
      ** Message data compression list
       15  MQCD-MSGCOMPLIST PIC S9(9) BINARY.
      ** Channel rank
       15  MQCD-CLWLCHANNELRANK PIC S9(9) BINARY.
      ** Channel priority
       15  MQCD-CLWLCHANNELPRIORITY PIC S9(9) BINARY.
      ** Channel weight
       15  MQCD-CLWLCHANNELWEIGHT PIC S9(9) BINARY.
      ** Channel monitoring
       15  MQCD-CHANNELMONITORING PIC S9(9) BINARY.
      ** Channel statistics
       15  MQCD-CHANNELSTATISTICS PIC S9(9) BINARY.
      ** Ver:8 **
      ** Limit on sharing conversations
       15  MQCD-SHARINGCONVERSATIONS PIC S9(9) BINARY.
      ** Message property control
       15  MQCD-PROPERTYCONTROL PIC S9(9) BINARY.
      ** Limit on SVRCONN channel instances
       15  MQCD-MAXINSTANCES PIC S9(9) BINARY.
      ** Limit on SVRCONN channel instances per client
       15  MQCD-MAXINSTANCESPERCLIENT PIC S9(9) BINARY.
      ** Client channel weight
       15  MQCD-CLIENTCHANNELWEIGHT PIC S9(9) BINARY.
      ** Connection affinity
       15  MQCD-CONNECTIONAFFINITY PIC S9(9) BINARY.
      ** Ver:9 **
      ** Batch data limit
       15  MQCD-BATCHDATALIMIT PIC S9(9) BINARY.
      ** Use Dead Letter Queue
       15  MQCD-USEDLQ PIC S9(9) BINARY.
      ** Default client reconnect option
       15  MQCD-DEFRECONNECT PIC S9(9) BINARY.
      ** Ver:10 **
      ** Certificate Label
       15 MQCD-CERTLABL PIC X (64)
      ** Ver:11 **
      ** AMS Security policy protection
       15 MQCD-SPLPROTECTION PIC S9(9) BINARY
      ** Ver:12 **
```

Deklaracja RPG (ILE)

Ta deklaracja jest deklaracją języka RPG dla struktury MQCD.

```
D* MQCD Structure
     D*
     D* Channel definition name
     D  CDCHN                  1     20
     D* Structure version number
     D  CDVER                 21     24I 0
     D* Channel type
     D  CDCHT                 25     28I 0
     D* Transport type
     D  CDTRT                 29     32I 0
     D* Channel description
     D  CDDES                 33     96
     D* Queue manager name
     D  CDQM                  97    144
     D* Transmission queue name
     D  CDXQ                 145    192
     D* First 20 bytes of connection name
     D  CDSCN                193    212
```

```
     D* Reserved
     D  CDMCA                213    232
     D* LU 6.2 Mode name
     D  CDMOD                233    240
     D* LU 6.2 transaction program name
     D  CDTP                 241    304
     D* Batch size
     D  CDBS                 305    308I 0
     D* Disconnect interval
     D  CDDI                 309    312I 0
     D* Short retry count
     D  CDSRC                313    316I 0
     D* Short retry wait interval
     D  CDSRI                317    320I 0
     D* Long retry count
     D  CDLRC                321    324I 0
     D* Long retry wait interval
     D  CDLRI                325    328I 0
     D* Channel security exit name
     D  CDSCX                329    348
     D* Channel message exit name
     D  CDMSX                349    368
     D* Channel send exit name
     D  CDSNX                369    388
     D* Channel receive exit name
     D  CDRCX                389    408
     D* Highest allowable message sequence number
     D  CDSNW                409    412I 0
     D* Maximum message length
     D  CDMML                413    416I 0
     D* Put authority
     D  CDPA                 417    420I 0
     D* Data conversion
     D  CDDC                 421    424I 0
     D* Channel security exit user data
     D  CDSCD                425    456
     D* Channel message exit user data
     D  CDMSD                457    488
     D* Channel send exit user data
     D  CDSND                489    520
     D* Channel receive exit user data
     D  CDRCD                521    552
     D* Ver:1 **
     D* User identifier
     D  CDUID                553    564
     D* Password
     D  CDPW                 565    576
     D* First 12 bytes of MCA user identifier
     D  CDAUI                577    588
     D* Message channel agent type
     D  CDCAT                589    592I 0
     D* Connection name
     D  CDCON                593    848
     D  CDCN2                849    856
     D* First 12 bytes of user identifier from partner
     D  CDRUI                857    868
     D* Password from partner
     D  CDRPW                869    880
     D* Ver:2 **
     D* Channel message retry exit name
     D  CDMRX                881    900
     D* Channel message retry exit user data
     D  CDMRD                901    932
     D* Number of times MCA will try to put the message, after first
     D* attempt has failed
     D  CDMRC                933    936I 0
     D* Minimum interval in milliseconds after which the open or put
     D* operation will be retried
     D  CDMRI                937    940I 0
     D* Ver:3 **
     D* Time in seconds between heartbeat flows
     D  CDHBI                941    944I 0
     D* Batch duration
     D  CDBI                 945    948I 0
     D* Speed at which nonpersistent messages are sent
     D  CDNPM                949    952I 0
     D* Length of MQCD structure
     D  CDLEN                953    956I 0
     D* Length of exit name
     D  CDXNL                957    960I 0
     D* Length of exit user data
     D  CDXDL                961    964I 0
```

```
     D* Number of message exits defined
     D  CDMXD                 965    968I 0
     D* Number of send exits defined
     D  CDSXD                 969    972I 0
     D* Number of receive exits defined
     D  CDRXD                 973    976I 0
     D* Address of first MsgExit field
     D  CDMXP                 977    992*
     D* Address of first MsgUserData field
     D  CDMUP                 993   1008*
     D* Address of first SendExit field
     D  CDSXP                1009   1024*
     D* Address of first SendUserData field
     D  CDSUP                1025   1040*
     D* Address of first ReceiveExit field
     D  CDRXP                1041   1056*
     D* Address of first ReceiveUserData field
     D  CDRUP                1057   1072*
     D* Ver:4 **
     D* Address of a list of cluster names
     D  CDCLP                1073   1088*
     D* Number of clusters to which the channel belongs
     D  CDCLD                1089   1092I 0
     D* Network priority
     D  CDNP                 1093   1096I 0
     D* Ver:5 **
     D* Length of long MCA user identifier
     D  CDLML                1097   1100I 0
     D* Length of long remote user identifier
     D  CDLRL                1101   1104I 0
     D* Address of long MCA user identifier
     D  CDLMP                1105   1120*
     D* Address of long remote user identifier
     D  CDLRP                1121   1136*
     D* MCA security identifier
     D  CDMSI                1137   1176
     D* Remote security identifier
     D  CDRSI                1177   1216
     D* Ver:6 **
     D* TLS CipherSpec
     D  CDSCS                1217   1248
     D* Address of TLS peer name
     D  CDSPN                1249   1264*
     D* Length of TLS peer name
     D  CDSPL                1265   1268I 0
     D* Whether TLS client authentication is required
     D  CDSCA                1269   1272I 0
     D* Keepalive interval
     D  CDKAI                1273   1276I 0
     D* Local communications address
     D  CDLOA                1277   1324
     D* Batch heartbeat interval
     D  CDBHB                1325   1328I 0
     D* Ver:7 **
     D* Header data compression list
     D  CDHCL0
     D  CDHCL1               1329   1332I 0
     D  CDHCL2               1333   1336I 0
     D  CDHCL                        10I 0 DIM(2) OVERLAY(CDHCL0)
     D* Message data compression list
     D  CDMCL0
     D  CDMCL1               1337   1340I 0
     D  CDMCL2               1341   1344I 0
     D  CDMCL3               1345   1348I 0
     D  CDMCL4               1349   1352I 0
     D  CDMCL5               1353   1356I 0
     D  CDMCL6               1357   1360I 0
     D  CDMCL7               1361   1364I 0
     D  CDMCL8               1365   1368I 0
     D  CDMCL9               1369   1372I 0
     D  CDMCL10              1373   1376I 0
     D  CDMCL11              1377   1380I 0
     D  CDMCL12              1381   1384I 0
     D  CDMCL13              1385   1388I 0
     D  CDMCL14              1389   1392I 0
     D  CDMCL15              1393   1396I 0
     D  CDMCL16              1397   1400I 0
     D  CDMCL                        10I 0 DIM(16) OVERLAY(CDMCL0)
     D* Channel rank
     D  CDCWCR               1401   1404I 0
     D* Channel priority
     D  CDCWCP               1405   1408I 0
```

```
     D* Channel weight
     D  CDCWCW              1409   1412I 0
     D* Channel monitoring
     D  CDCHLMON            1413   1416I 0
     D* Channel statistics
     D  CDCHLST             1417   1420I 0
     D* Ver:8 **
     D* Limit on sharing conversations
     D  CDSHC               1421   1424I 0
     D* Message property control
     D  CDPRC               1425   1428I 0
     D* Limit on SVRCONN channel instances
     D  CDMXIN              1429   1432I 0
     D* Limit on SVRCONN channel instances per client
     D  CDMXIC              1433   1436I 0
     D* Client channel weight
     D  CDCLNCHLW           1437   1440I 0
     D* Connection affinity
     D  CDCONNAFF           1441   1444I 0
     D* Ver:9 **
     D* Batch data limit
     D  CDBDL               1445   1448I 0
     D* Use Dead Letter Queue
     D  CDUDLQ              1449   1452I 0
     D* Default client reconnect option
     D  CDDRCN              1453   1456I 0
     D* Ver:10 **
```

Deklaracja asemblera System/390

Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQCD.

```
MQCD                          DSECT
MQCD_CHANNELNAME              DS   CL20   Channel definition name
MQCD_VERSION                  DS   F      Structure version number
MQCD_CHANNELTYPE              DS   F      Channel type
MQCD_TRANSPORTTYPE            DS   F      Transport type
MQCD_DESC                     DS   CL64   Channel description
MQCD_QMGRNAME                 DS   CL48   Queue manager name
MQCD_XMITQNAME                DS   CL48   Transmission queue name
MQCD_SHORTCONNECTIONNAME      DS   CL20   First 20 bytes of connection
*                                         name
MQCD_MCANAME                  DS   CL20   Reserved
MQCD_MODENAME                 DS   CL8    LU 6.2 Mode name
MQCD_TPNAME                   DS   CL64   LU 6.2 transaction program name
MQCD_BATCHSIZE                DS   F      Batch size
MQCD_DISCINTERVAL             DS   F      Disconnect interval
MQCD_SHORTRETRYCOUNT          DS   F      Short retry count
MQCD_SHORTRETRYINTERVAL       DS   F      Short retry wait interval
MQCD_LONGRETRYCOUNT           DS   F      Long retry count
MQCD_LONGRETRYINTERVAL        DS   F      Long retry wait interval
MQCD_SECURITYEXIT             DS   CLn    Channel security exit name
MQCD_MSGEXIT                  DS   CLn    Channel message exit name
MQCD_SENDEXIT                 DS   CLn    Channel send exit name
MQCD_RECEIVEEXIT              DS   CLn    Channel receive exit name
MQCD_SEQNUMBERWRAP            DS   F      Highest allowable message
*                                         sequence number
MQCD_MAXMSGLENGTH             DS   F      Maximum message length
MQCD_PUTAUTHORITY             DS   F      Put authority
MQCD_DATACONVERSION           DS   F      Data conversion
MQCD_SECURITYUSERDATA         DS   CL32   Channel security exit user data
MQCD_MSGUSERDATA              DS   CL32   Channel message exit user data
MQCD_SENDUSERDATA             DS   CL32   Channel send exit user data
MQCD_RECEIVEUSERDATA          DS   CL32   Channel receive exit user data
MQCD_USERIDENTIFIER           DS   CL12   User identifier
MQCD_PASSWORD                 DS   CL12   Password
MQCD_MCAUSERIDENTIFIER        DS   CL12   First 12 bytes of MCA user
*                                         identifier
MQCD_MCATYPE                  DS   F      Message channel agent type
MQCD_CONNECTIONNAME           DS   CL264  Connection name
MQCD_REMOTEUSERIDENTIFIER     DS   CL12   First 12 bytes of user
*                                         identifier from partner
MQCD_REMOTEPASSWORD           DS   CL12   Password from partner
MQCD_MSGRETRYEXIT             DS   CLn    Channel message retry exit name
MQCD_MSGRETRYUSERDATA         DS   CL32   Channel message retry exit user
*                                         data
MQCD_MSGRETRYCOUNT            DS   F      Number of times MCA will try to
*                                         put the message, after the
*                                         first attempt has failed
```

```
MQCD_MSGRETRYINTERVAL        DS   F      Minimum interval in
*                                        milliseconds after which the
*                                        open or put operation will be
*                                        retried
MQCD_HEARTBEATINTERVAL       DS   F      Time in seconds between
*                                        heartbeat flows
MQCD_BATCHINTERVAL           DS   F      Batch duration
MQCD_NONPERSISTENTMSGSPEED   DS   F      Speed at which nonpersistent
*                                        messages are sent
MQCD_STRUCLENGTH             DS   F      Length of MQCD structure
MQCD_EXITNAMELENGTH          DS   F      Length of exit name
MQCD_EXITDATALENGTH          DS   F      Length of exit user data
MQCD_MSGEXITSDEFINED         DS   F      Number of message exits defined
MQCD_SENDEXITSDEFINED        DS   F      Number of send exits defined
MQCD_RECEIVEEXITSDEFINED     DS   F      Number of receive exits defined
MQCD_MSGEXITPTR              DS   F      Address of first MSGEXIT field
MQCD_MSGUSERDATAPTR          DS   F      Address of first MSGUSERDATA
*                                        field
MQCD_SENDEXITPTR             DS   F      Address of first SENDEXIT field
MQCD_SENDUSERDATAPTR         DS   F      Address of first SENDUSERDATA
*                                        field
MQCD_RECEIVEEXITPTR          DS   F      Address of first RECEIVEEXIT
*                                        field
MQCD_RECEIVEUSERDATAPTR      DS   F      Address of first
*                                        RECEIVEUSERDATA field
MQCD_CLUSTERPTR              DS   F      Address of a list of cluster
*                                        names
MQCD_CLUSTERSDEFINED         DS   F      Number of clusters to which the
*                                        channel belongs
MQCD_NETWORKPRIORITY         DS   F      Network priority
MQCD_LONGMCAUSERIDLENGTH     DS   F      Length of long MCA user
*                                        identifier
MQCD_LONGREMOTEUSERIDLENGTH  DS   F      Length of long remote user
*                                        identifier
MQCD_LONGMCAUSERIDPTR        DS   F      Address of long MCA user
*                                        identifier
MQCD_LONGREMOTEUSERIDPTR     DS   F      Address of long remote user
*                                        identifier
MQCD_MCASECURITYID           DS   XL40   MCA security identifier
MQCD_REMOTESECURITYID        DS   XL40   Remote security identifier
MQCD_SSLCIPHERSPEC           DS   CL32   TLS CipherSpec
MQCD_SSLPEERNAMEPTR          DS   F      Address of TLS peer name
MQCD_SSLPEERNAMELENGTH       DS   F      Length of TLS peer name
MQCD_SSLCLIENTAUTH           DS   F      Whether TLS client
*                                        authentication is required
MQCD_KEEPALIVEINTERVAL       DS   F      Keepalive interval
MQCD_LOCALADDRESS            DS   CL48   Local communications address
MQCD_BATCHHEARTBEAT          DS   F      Batch heartbeat interval
MQCD_HDRCOMPLIST             DS   CL2    Header data compression list
MQCD_MSGCOMPLIST             DS   CL16   Message data compression list
MQCD_CLWLCHANNELRANK         DS   F      Channel rank
MQCD_CLWLCHANNELPRIORITY     DS   F      Channel priority
MQCD_CLWLCHANNELWEIGHT       DS   F      Channel weight
MQCD_CHANNELMONITORING       DS   F      Channel monitoring
MQCD_CHANNELSTATISTICS       DS   F      Channel statistics
MQCD_SHARINGCONVERSATIONS    DS   F      Limit on sharing
*                                        conversations
MQCD_PROPERTYCONTROL         DS   F      Message property
*                                        control
MQCD_SHARINGCONVERSATIONS    DS   F      Limit on sharing conversations
MQCD_PROPERTYCONTROL         DS   F      Message property control
MQCD_MAXINSTANCES            DS   F      Limit on SVRCONN chl instances
MQCD_MAXINSTANCESPERCLIENT   DS   F      Limit on SVRCONN chl instances
                                         per client
MQCD_CLIENTCHANNELWEIGHT     DS   F      Channel weight
MQCD_CONNECTIONAFFINITY      DS   F      Connection Affinty
MQCD_BATCHDATALIMIT          DS   F      Batch data limit
MQCD_USEDLQ                  DS   F      Use dead-letter queue
MQCD_DEFRECONNECT            DS   F      Default client reconnect option
MQCD_CERTLABL                DS   F      Certificate label
MQCD_SPLPROTECTION           DS   F      AMS Security policy protection
MQCD_LENGTH                  EQU  *-MQCD
                             ORG  MQCD
MQCD_AREA                    DS   CL(MQCD_LENGTH)
```

Deklaracja Visual Basic

Ta deklaracja jest deklaracją Visual Basic struktury MQCD.

W języku Visual Basic struktura MQCD może być używana ze strukturą MQCNO w wywołaniu MQCONNX.

```
Type MQCD
  ChannelName            As String*20  'Channel definition name'
  Version                As Long       'Structure version number'
  ChannelType            As Long       'Channel type'
  TransportType          As Long       'Transport type'
  Desc                   As String*64  'Channel description'
  QMgrName               As String*48  'Queue manager name'
  XmitQName              As String*48  'Transmission queue name'
  ShortConnectionName    As String*20  'First 20 bytes of connection'
                                       'name'
  MCAName                As String*20  'Reserved'
  ModeName               As String*8   'LU 6.2 Mode name'
  TpName                 As String*64  'LU 6.2 transaction program name'
  BatchSize              As Long       'Batch size'
  DiscInterval           As Long       'Disconnect interval'
  ShortRetryCount        As Long       'Short retry count'
  ShortRetryInterval     As Long       'Short retry wait interval'
  LongRetryCount         As Long       'Long retry count'
  LongRetryInterval      As Long       'Long retry wait interval'
  SecurityExit           As String*128 'Channel security exit name'
  MsgExit                As String*128 'Channel message exit name'
  SendExit               As String*128 'Channel send exit name'
  ReceiveExit            As String*128 'Channel receive exit name'
  SeqNumberWrap          As Long       'Highest allowable message'
                                       'sequence number'
  MaxMsgLength           As Long       'Maximum message length'
  PutAuthority           As Long       'Put authority'
  DataConversion         As Long       'Data conversion'
  SecurityUserData       As String*32  'Channel security exit user data'
  MsgUserData            As String*32  'Channel message exit user data'
  SendUserData           As String*32  'Channel send exit user data'
  ReceiveUserData        As String*32  'Channel receive exit user data'
  UserIdentifier         As String*12  'User identifier'
  Password               As String*12  'Password'
  MCAUserIdentifier      As String*12  'First 12 bytes of MCA user'
                                       'identifier'
  MCAType                As Long       'Message channel agent type'
  ConnectionName         As String*264 'Connection name'
  RemoteUserIdentifier   As String*12  'First 12 bytes of user'
                                       'identifier from partner'
  RemotePassword         As String*12  'Password from partner'
  MsgRetryExit           As String*128 'Channel message retry exit name'
  MsgRetryUserData       As String*32  'Channel message retry exit user'
                                       'data'
  MsgRetryCount          As Long       'Number of times MCA will try to'
                                       'put the message, after the'
                                       'first attempt has failed'
  MsgRetryInterval       As Long       'Minimum interval in'
                                       'milliseconds after which the'
                                       'open or put operation will be'
                                       'retried'
  HeartbeatInterval      As Long       'Time in seconds between'
                                       'heartbeat flows'
  BatchInterval          As Long       'Batch duration'
  NonPersistentMsgSpeed  As Long       'Speed at which nonpersistent'
                                       'messages are sent'
  StrucLength            As Long       'Length of MQCD structure'
  ExitNameLength         As Long       'Length of exit name'
  ExitDataLength         As Long       'Length of exit user data'
  MsgExitsDefined        As Long       'Number of message exits defined'
  SendExitsDefined       As Long       'Number of send exits defined'
  ReceiveExitsDefined    As Long       'Number of receive exits defined'
  MsgExitPtr             As MQPTR      'Address of first MsgExit field'
  MsgUserDataPtr         As MQPTR      'Address of first MsgUserData'
                                       'field'
  SendExitPtr            As MQPTR      'Address of first SendExit field'
  SendUserDataPtr        As MQPTR      'Address of first SendUserData'
                                       'field'
  ReceiveExitPtr         As MQPTR      'Address of first ReceiveExit'
                                       'field'
  ReceiveUserDataPtr     As MQPTR      'Address of first'
                                       'ReceiveUserData field'
  ClusterPtr             As MQPTR      'Address of a list of cluster'
                                       'names'
  ClustersDefined        As Long       'Number of clusters to which the'
                                       'channel belongs'
  NetworkPriority        As Long       'Network priority'
  LongMCAUserIdLength    As Long       'Length of long MCA user'
                                       'identifier'
  LongRemoteUserIdLength As Long       'Length of long remote user'
```

```
                                    'identifier'
  LongMCAUserIdPtr       As MQPTR   'Address of long MCA user'
                                    'identifier'
  LongRemoteUserIdPtr    As MQPTR   'Address of long remote user'
                                    'identifier'
  MCASecurityId          As MQBYTE40 'MCA security identifier'
  RemoteSecurityId       As MQBYTE40 'Remote security identifier'
  SSLCipherSpec          As String*32 'TLS CipherSpec'
  SSLPeerNamePtr         As MQPTR   'Address of TLS peer name'
  SSLPeerNameLength      As Long    'Length of TLS peer name'
  SSLClientAuth          As Long    'Whether TLS client'
                                    'authentication is required'
  KeepAliveInterval      As Long    'Keepalive interval'
  LocalAddress           As String*48 'Local communications address'
  BatchHeartbeat         As Long    'Batch heartbeat interval'
  HdrCompList(0 to 1)    As Long2   'Header data compression list'
  MsgCompList(0 To 15)   As Long16  'Message data compression list'
  CLWLChannelRank        As Long    'Channel Rank'
  CLWLChannelPriority    As Long    'Channel priority'
  CLWLChannelWeight      As Long    'Channel Weight'
  ChannelMonitoring      As Long    'Channel Monitoring control'
  ChannelStatistics      As Long    'Channel Statistics'
End Type
```

Zmiana pól MQCD w wyjściu kanału

Wyjście kanału może zmieniać pola w MQCD. Jednak zmiany te zwykle nie są wykonywane, z wyjątkiem wymienionych okoliczności.

Jeśli program obsługi wyjścia kanału zmieni pole w strukturze danych MQCD, nowa wartość jest zwykle ignorowana przez proces kanału IBM MQ . Jednak nowa wartość pozostaje w MQCD i jest przekazywana do wszystkich pozostałych wyjść w łańcuchu wyjścia i do każdej konwersacji współużytkującej instancję kanału.

Jeśli właściwość SharingConversations jest ustawiona na wartość FALSE w strukturze MQCXP, zmiany w niektórych polach mogą być wykonywane w zależności od typu programu obsługi wyjścia, typu kanału i kodu przyczyny wyjścia. W poniższej tabeli przedstawiono pola, które można zmieniać i wpływać na zachowanie kanału, a także okoliczności, w jakich. Jeśli program obsługi wyjścia zmieni jedno z tych pól w innych okolicznościach lub inne pole, nowa wartość zostanie zignorowana przez proces kanału. Nowa wartość pozostaje w MQCD i jest przekazywana do wszystkich pozostałych wyjść w łańcuchu wyjścia i do każdej konwersacji współużytkującej instancję kanału.

Dowolny typ programu obsługi wyjścia po wywołaniu w celu zainicjowania (MQXR_INIT) może zmienić pole ChannelName dowolnego typu kanału, o ile opcja SharingConverstions dla MQCXP ma wartość FALSE. Tylko wyjście zabezpieczeń może zmienić pole MCAUserIdentifier , niezależnie od wartości MQCXP SharingConverstions.

Tabela 823. Pola, które mogą być zmieniane i wpływać na zachowanie kanału

Pole	**Kod przyczyny wyjścia**	**Typ wyjścia**	**Typ kanału**
ChannelName	MQXR_INIT	Wszystkie	Wszystkie
TransportType	MQXR_INIT	Wszystkie	Wszystkie
XmitQName	MQXR_INIT	Wszystkie	SDR, RCVR
ModeName	MQXR_INIT	Wszystkie	Wszystkie
TpName	MQXR_INIT	Wszystkie	Wszystkie
BatchSize	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR

Tabela 823. Pola, które mogą być zmieniane i wpływać na zachowanie kanału (kontynuacja)			
Pole	**Kod przyczyny wyjścia**	**Typ wyjścia**	**Typ kanału**
DiscInterval	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Liczba ShortRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Odstęp czasu ShortRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Liczba LongRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
Odstęp czasu LongRetry	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SeqNumberZawijaj	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MaxMsg	MQXR_INIT	Wszystkie	Wszystkie
PutAuthority	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
DataConversion	MQXR_INIT	Wszystkie	Wszystkie
MCAUserIdentifier	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Zabezpieczenia	RCVR, RQSTR, SVRCONN, CLUSRCVR
ConnectionName	MQXR_INIT	Wszystkie	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR

Tabela 823. Pola, które mogą być zmieniane i wpływać na zachowanie kanału (kontynuacja)

Pole	Kod przyczyny wyjścia	Typ wyjścia	Typ kanału
MsgRetryUserData	MQXR_INIT	Wszystkie	RCVR, RQSTR, CLUSRCVR
MsgRetryLiczba	MQXR_INIT	Wszystkie	RCVR, RQSTR, CLUSRCVR
Odstęp czasu między ponownymi próbami MsgRetry	MQXR_INIT	Wszystkie	RCVR, RQSTR, CLUSRCVR
HeartbeatInterval	MQXR_INIT	Wszystkie	Wszystkie
BatchInterval	MQXR_INIT	Wszystkie	SDR, SVR, CLUSSDR, CLUSRCVR
NonPersistentMsgSpeed	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
MCASecurityId	MQXR_INIT, MQXR_INIT_SEC, MQXR_SEC_MSG, MQXR_SEC_PARMS	Zabezpieczenia	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
SSLCipherSpec	MQXR_INIT	Wszystkie	Wszystkie
SSLPeerNamePtr	MQXR_INIT	Wszystkie	Wszystkie
Długość SSLPeerName	MQXR_INIT	Wszystkie	Wszystkie
SSLClientAuth	MQXR_INIT	Wszystkie	SVR, RCVR, RQSTR, SVRCONN, CLUSRCVR
Przedział czasu KeepAlive	MQXR_INIT	Wszystkie	Wszystkie
LocalAddress	MQXR_INIT	Wszystkie	SDR, SVR, RQSTR, CLNTCONN, CLUSSDR, CLUSRCVR
BatchHeartbeat	MQXR_INIT	Wszystkie	SDR, SVR, CLUSSDR, CLUSRCVR
Lista HdrComp	MQXR_INIT	Wszystkie	Wszystkie
Lista MsgComp	MQXR_INIT	Wszystkie	Wszystkie

Tabela 823. Pola, które mogą być zmieniane i wpływać na zachowanie kanału (kontynuacja)			
Pole	**Kod przyczyny wyjścia**	**Typ wyjścia**	**Typ kanału**
ChannelMonitoring	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, SVRCONN, CLUSSDR, CLUSRCVR
ChannelStatistics	MQXR_INIT	Wszystkie	SDR, SVR, RCVR, RQSTR, CLUSSDR, CLUSRCVR
SharingConversations	MQXR_INIT	Wszystkie	SVRCONN, CLNTCONN
PropertyControl	MQXR_INIT	Wszystkie	SDR, SVR, CLUSSDR, CLUSRCVR

MQCXP-parametr wyjścia kanału

Struktura MQCXP jest przekazywana do każdego typu wyjścia wywoływanego przez agent kanału komunikatów (MCA), kanał połączenia z klientem lub kanał połączenia z serwerem.

Patrz MQ_CHANNEL_EXIT.

Pola opisane w kolejnych opisach jako "wejście do wyjścia" są ignorowane przez kanał, gdy wyjście zwraca sterowanie do kanału. Wszystkie pola wejściowe, które zostaną zmienione w bloku parametru wyjścia kanału, nie zostaną zachowane do następnego wywołania. Zmiany wprowadzone w polach wejściowych i wyjściowych (na przykład w polu *ExitUserArea*) są zachowywane tylko dla wywołań tej instancji wyjścia. Takich zmian nie można używać do przekazywania danych między różnymi wyjściami zdefiniowanymi w tym samym kanale lub między tymi samymi wyjściami zdefiniowanymi w różnych kanałach.

Odsyłacze pokrewne

“Pola” na stronie 1568
W tym temacie opisano wszystkie pola w strukturze MQCXP oraz opisano poszczególne pola.

“Deklaracja C” na stronie 1580
Ta deklaracja jest deklaracją języka C dla struktury MQCXP.

“Deklaracja języka COBOL” na stronie 1581
Ta deklaracja jest deklaracją języka COBOL dla struktury MQCXP.

“Deklaracja RPG (ILE)” na stronie 1581
Ta deklaracja jest deklaracją języka RPG dla struktury MQCXP.

“Deklaracja asemblera System/390” na stronie 1582
Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQCXP.

Pola

W tym temacie opisano wszystkie pola w strukturze MQCXP oraz opisano poszczególne pola.

StrucId (MQCHAR4)
To pole określa identyfikator struktury.

Wartość musi być następująca:

MQCXP_STRUC_ID
Identyfikator struktury parametru wyjścia kanału.

Dla języka programowania C zdefiniowana jest również stała MQCXP_STRUC_ID_ARRAY; ta stała ma taką samą wartość jak MQCXP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe do wyjścia.

Wersja (MQLONG)
To pole określa numer wersji struktury.

Wartość zależy od środowiska:

MQCXP_VERSION_1
Struktura parametru wyjścia kanału Version-1 .

MQCXP_VERSION_2
Struktura parametru wyjścia kanału Version-2 .

MQCXP_VERSION_3
Struktura parametru wyjścia kanału Version-3 .

Linux AIX Pole ma tę wartość w systemach AIX and Linux , które nie są wymienione w innym miejscu.

MQCXP_VERSION_4
Struktura parametru wyjścia kanału Version-4 .

MQCXP_VERSION_5
Struktura parametru wyjścia kanału Version-5 .

MQCXP_VERSION_6
Struktura parametru wyjścia kanału Version-6 .

MQCXP_VERSION_8
Struktura parametru wyjścia kanału Version-8 .

z/OS Pole ma tę wartość w pliku z/OS.

MQCXP_VERSION_9
Struktura parametru wyjścia kanału Version-9 .

Pole ma tę wartość w następujących środowiskach:

- AIX AIX
- IBM i IBM i
- Linux Linux
- Windows Windows
- z/OS z/OS

Pola, które istnieją tylko w nowszych wersjach struktury, są identyfikowane jako takie w opisach pól. Następująca stała określa numer wersji bieżącej:

MQCXP_CURRENT_VERSION
Bieżąca wersja struktury parametru wyjścia kanału.

Wartość zależy od środowiska.

Uwaga: Po wprowadzeniu nowej wersji struktury MQCXP układ istniejącej części nie ulega zmianie. Dlatego wyjście musi sprawdzić, czy numer wersji jest równy lub większy od najniższej wersji, która zawiera pola, które muszą być używane przez wyjście.

Jest to pole wejściowe do wyjścia.

ExitId (MQLONG)
To pole określa typ wywoływanego wyjścia i jest ustawiane na wejściu do procedury wyjścia.

Dozwolone są następujące wartości:

MQXT_CHANNEL_SEC_EXIT
Wyjście zabezpieczeń kanału.

MQXT_CHANNEL_MSG_EXIT
Wyjście komunikatu kanału.

MQXT_CHANNEL_SEND_EXIT
Wyjście wysyłania kanału.

MQXT_CHANNEL_RCV_EXIT
Wyjście odbierania kanału.

MQXT_CHANNEL_MSG_RETRY_EXIT
Komunikat kanału-wyjście ponowienia.

MQXT_CHANNEL_AUTO_DEF_EXIT
Wyjście automatycznej definicji kanału.

W systemie z/OSten typ wyjścia jest obsługiwany tylko dla kanałów typu MQCHT_CLUSSDR i MQCHT_CLUSRCVR.

Jest to pole wejściowe do wyjścia.

ExitReason (MQLONG)
To pole określa przyczynę wywołania wyjścia i jest ustawione na wejściu do procedury wyjścia.

Nie jest on używany przez wyjście automatycznej definicji. Dozwolone są następujące wartości:

MQXR_INIT
Inicjowanie wyjścia.

Ta wartość wskazuje, że wyjście jest wywoływane po raz pierwszy. Umożliwia on wyjściu uzyskanie i zainicjowanie wszystkich potrzebnych zasobów (na przykład pamięci).

MQXR_TERM
Zakończenie pracy.

Ta wartość wskazuje, że wyjście zostanie zakończone. Wyjście powinno zwolnić wszystkie zasoby, które uzyskało od czasu zainicjowania (na przykład: pamięć).

MQXR_MSG
Przetwarzanie komunikatu.

Ta wartość wskazuje, że wyjście jest wywoływane w celu przetworzenia komunikatu. Ta wartość występuje tylko dla wyjść komunikatów kanału.

MQXR_XMIT
Przetwarzanie transmisji.

Ta wartość występuje tylko dla wyjść wysyłania i odbierania kanału.

MQXR_SEC_MSG
Odebrano komunikat bezpieczeństwa.

Ta wartość występuje tylko dla wyjść zabezpieczeń kanału.

MQXR_INIT_SEK
Inicjowanie wymiany zabezpieczeń.

Ta wartość występuje tylko dla wyjść zabezpieczeń kanału.

Wyjście zabezpieczeń odbiornika jest zawsze wywoływane z tej przyczyny natychmiast po wywołaniu za pomocą MQXR_INIT, aby umożliwić mu zainicjowanie wymiany zabezpieczeń. Jeśli odrzuci możliwość (zwracając MQXCC_OK zamiast MQXCC_SEND_SEC_MSG

lub MQXCC_SEND_AND_REQUEST_SEC_MSG), wyjście zabezpieczeń nadawcy jest wywoływane z MQXR_INIT_SEC.

Jeśli wyjście zabezpieczeń odbiornika nie inicjuje wymiany zabezpieczeń (przez zwrócenie MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG), wyjście zabezpieczeń nadawcy nie jest nigdy wywoływane z MQXR_INIT_SEC; zamiast tego jest wywoływane z MQXR_SEC_MSG w celu przetworzenia komunikatu odbiornika. (W obu przypadkach jest on najpierw wywoływany z opcją MQXR_INIT).

Jeśli jedno z wyjść zabezpieczeń nie wymaga zakończenia kanału (przez ustawienie parametru *ExitResponse* na wartość MQXCC_SUPPRESS_FUNCTION lub MQXCC_CLOSE_CHANNEL), wymiana zabezpieczeń musi zostać zakończona po stronie, która zainicjowała wymianę. Dlatego, jeśli wyjście zabezpieczeń zostanie wywołane z MQXR_INIT_SEC i nie zainicjuje wymiany, przy następnym wywołaniu wyjścia będzie ono z MQXR_SEC_MSG. Dzieje się tak niezależnie od tego, czy istnieje komunikat bezpieczeństwa dla wyjścia do przetworzenia, czy nie. Jeśli partner zwróci komunikat MQXCC_SEND_SEC_MSG lub MQXCC_SEND_AND_REQUEST_SEC_MSG, ale nie zwróci komunikatu MQXCC_OK lub nie ma wyjścia zabezpieczeń dla partnera, zostanie wyświetlony komunikat bezpieczeństwa. Jeśli nie ma komunikatu bezpieczeństwa do przetworzenia, wyjście zabezpieczeń na końcu inicjującym jest ponownie wywoływane z wartością *DataLength* równą zero.

MQXR_RETRY
Ponów komunikat.

Ta wartość występuje tylko dla wyjść ponowień komunikatu.

MQXR_AUTO_CLUSSDR
Automatyczna definicja kanału nadawczego klastra.

Ta wartość występuje tylko w przypadku wyjść automatycznego definiowania kanału.

MQXR_AUTO_RECEIVER
Automatyczna definicja kanału odbiorczego.

Ta wartość występuje tylko w przypadku wyjść automatycznego definiowania kanału.

MQXR_AUTO_SVRCONN
Automatyczna definicja kanału połączenia z serwerem.

Ta wartość występuje tylko w przypadku wyjść automatycznego definiowania kanału.

MQXR_AUTO_CLUSRCVR
Automatyczna definicja kanału odbiorczego klastra.

Ta wartość występuje tylko w przypadku wyjść automatycznego definiowania kanału.

PARMS MQXR_SEC_PARMS
Parametry zabezpieczeń

Ta wartość ma zastosowanie tylko do wyjść zabezpieczeń i wskazuje, że do wyjścia jest przekazywana struktura MQCSP. Więcej informacji na ten temat zawiera sekcja "MQCSP-parametry zabezpieczeń" na stronie 341.

Uwaga:

1. Jeśli dla kanału zdefiniowano więcej niż jedno wyjście, każde z nich jest wywoływane za pomocą MQXR_INIT podczas inicjowania agenta MCA. Ponadto są one wywoływane z parametrem MQXR_TERM, gdy agent MCA jest zakończony.
2. W przypadku wyjścia automatycznej definicji kanału parametr *ExitReason* nie jest ustawiany, jeśli wartość parametru *Version* jest mniejsza niż wartość parametru MQCXP_VERSION_4. W tym przypadku zakłada się wartość MQXR_AUTO_SVRCONN.

Jest to pole wejściowe do wyjścia.

ExitResponse (MQLONG)
To pole określa odpowiedź z wyjścia.

To pole jest ustawiane przez wyjście w celu komunikacji z agentem MCA. Musi to być jedna z następujących wartości:

MQXCC_OK
Wyjście zakończone pomyślnie.

- W przypadku wyjścia zabezpieczeń kanału ta wartość wskazuje, że przesyłanie komunikatów może być teraz kontynuowane w normalny sposób.
- Dla wyjścia ponowienia komunikatu kanału ta wartość wskazuje, że agent MCA musi czekać przez odstęp czasu zwrócony przez wyjście w polu *MsgRetryInterval* w produkcie MQCXP, a następnie ponowić komunikat.

Pole *ExitResponse2* może zawierać dodatkowe informacje.

MQXCC_SUPPRESS_FUNCTION,
Ukryj funkcję.

- Dla wyjścia zabezpieczeń kanału ta wartość wskazuje, że kanał musi zostać zakończony.
- W przypadku wyjścia komunikatu kanału ta wartość wskazuje, że komunikat nie ma być kontynuowany w kierunku miejsca docelowego. Zamiast tego agent MCA generuje komunikat raportu o wyjątku (jeśli został zażądany przez nadawcę oryginalnego komunikatu) i umieszcza komunikat zawarty w oryginalnym buforze w kolejce niedostarczonych komunikatów (jeśli nadawca określił parametr MQRO_DEAD_LETTER_Q) lub usuwa go (jeśli nadawca określił parametr MQRO_DISCARD_MSG).

  W przypadku trwałych komunikatów, jeśli nadawca określił parametr MQRO_DEAD_LETTER_Q, ale umieszczenie w kolejce niedostarczonych komunikatów nie powiodło się lub nie ma kolejki niedostarczonych komunikatów, oryginalny komunikat pozostaje w kolejce transmisji, a komunikat raportu nie jest generowany. Oryginalny komunikat pozostaje również w kolejce transmisji, jeśli nie można pomyślnie wygenerować komunikatu raportu.

  Pole *Feedback* w strukturze MQDLH na początku komunikatu w kolejce niedostarczonych komunikatów wskazuje, dlaczego komunikat został umieszczony w kolejce niedostarczonych komunikatów. Ten kod sprzężenia zwrotnego jest również używany w deskryptorze komunikatu raportu o wyjątkach (jeśli został zażądany przez nadawcę).
- W przypadku wyjścia dla ponowienia komunikatu kanału wartość ta wskazuje, że agent MCA nie czeka i nie próbuje ponownie komunikatu; zamiast tego agent MCA kontynuuje natychmiast normalne przetwarzanie niepowodzenia (komunikat jest umieszczany w kolejce niedostarczonych komunikatów lub odrzucany, zgodnie z określeniem nadawcy komunikatu).
- Dla wyjścia automatycznego definiowania kanału należy podać opcję MQXCC_OK lub MQXCC_SUPPRESS_FUNCTION. Jeśli żadna z tych wartości nie zostanie podana, domyślnie przyjmowana jest opcja MQXCC_SUPPRESS_FUNCTION, a definicja automatyczna jest porzucana.

Ta odpowiedź nie jest obsługiwana dla wyjść wysyłania i odbierania kanału.

MQXCC_SEND_SEC_MSG
Wyślij komunikat bezpieczeństwa.

Ta wartość może być ustawiona tylko przez wyjście zabezpieczeń kanału. Wskazuje, że wyjście udostępniło komunikat bezpieczeństwa, który musi zostać przesłany do partnera.

MQXCC_SEND_AND_REQUEST_SEC_MSG
Wyślij komunikat bezpieczeństwa, który wymaga odpowiedzi.

Ta wartość może być ustawiona tylko przez wyjście zabezpieczeń kanału. Wskazuje

- że wyjście dostarczyła komunikat bezpieczeństwa, który można przekazać partnerowi, oraz
- że wyjście wymaga odpowiedzi od partnera. Jeśli nie zostanie odebrana żadna odpowiedź, kanał musi zostać zakończony, ponieważ wyjście nie zdecydowało jeszcze, czy komunikacja może być kontynuowana.

MQXCC_SUPPRESS_EXIT
Pomiń wyjście.

- Ta wartość może być ustawiona przez wszystkie typy wyjścia kanału inne niż wyjście zabezpieczeń lub wyjście automatycznej definicji. Powoduje ona zablokowanie dalszego wywołania tego wyjścia (tak, jakby jego nazwa była pusta w definicji kanału) do momentu zakończenia kanału, gdy wyjście zostanie ponownie wywołane z parametrem *ExitReason* o wartości MQXR_TERM.
- Jeśli wyjście ponowienia komunikatu zwraca tę wartość, ponowienia komunikatu dla kolejnych komunikatów są kontrolowane przez atrybuty kanału *MsgRetryCount* i *MsgRetryInterval* w normalny sposób. Dla bieżącego komunikatu agent MCA wykonuje liczbę zaległych ponowień w odstępach czasu określonych przez atrybut kanału *MsgRetryInterval* , ale tylko wtedy, gdy kod przyczyny jest jednym z kodów, które agent MCA normalnie ponowiłby (patrz pole *MsgRetryCount* opisane w sekcji "MQCD-definicja kanału" na stronie 1526). Liczba zaległych ponowień jest wartością atrybutu **MsgRetryCount** pomniejszoną o liczbę ponowień, które zostały zwrócone przez wyjście MQXCC_OK dla bieżącego komunikatu. Jeśli ta liczba jest ujemna, agent MCA nie będzie ponawiał prób dla bieżącego komunikatu.

MQXCC_CLOSE_CHANNEL,
Zamknij kanał.

Ta wartość może być ustawiona przez dowolny typ wyjścia kanału z wyjątkiem wyjścia automatycznego definiowania.

Jeśli współużytkowanie konwersacji nie jest włączone, ta wartość powoduje zamknięcie kanału.

Jeśli współużytkowanie konwersacji jest włączone, ta wartość kończy konwersację. Jeśli ta konwersacja jest jedyną konwersacją w kanale, kanał również zostanie zamknięty.

To pole jest polem wejścia/wyjścia z wyjścia.

ExitResponse2 (MQLONG)
To pole określa odpowiedź dodatkową z wyjścia.

To pole jest ustawiane na zero przy wejściu do procedury zewnętrznej. Może on zostać ustawiony przez wyjście w celu udostępnienia dodatkowych informacji dla funkcji kanału IBM MQ . Nie jest on używany przez wyjście automatycznej definicji.

Wyjście może ustawić jedną lub więcej z następujących wartości. Jeśli wymagana jest więcej niż jedna, wartości są dodawane. Niepoprawne kombinacje są odnotowywane; inne kombinacje są dozwolone.

MQXR2_PUT_WITH_DEF_ACTION
Umieść z działaniem domyślnym.

Ta wartość jest ustawiana przez wyjście komunikatu kanału odbiorcy. Wskazuje, że komunikat ma zostać umieszczony z domyślnym działaniem agenta MCA, czyli domyślnym identyfikatorem użytkownika agenta MCA lub kontekstem *UserIdentifier* w deskryptorze MQMD (deskryptorze komunikatu) komunikatu.

Wartością jest zero, co odpowiada wartości początkowej ustawionej podczas wywoływania wyjścia. Stała jest udostępniana na potrzeby dokumentacji.

MQXR2_PUT_WITH_DEF_USERID
Umieść z domyślnym identyfikatorem użytkownika.

Tę wartość można ustawić tylko przez wyjście komunikatu kanału odbiorcy. Wskazuje, że komunikat ma zostać umieszczony z domyślnym identyfikatorem użytkownika agenta MCA.

MQXR2_PUT_WITH_MSG_USERID
Umieść z identyfikatorem użytkownika komunikatu.

Tę wartość można ustawić tylko przez wyjście komunikatu kanału odbiorcy. Wskazuje, że komunikat ma zostać umieszczony z kontekstem *UserIdentifier* w deskryptorze MQMD (deskryptor komunikatu) komunikatu (mógł zostać zmodyfikowany przez wyjście).

Należy ustawić tylko jedną z następujących wartości: MQXR2_PUT_WITH_DEF_ACTION, MQXR2_PUT_WITH_DEF_USERIDi MQXR2_PUT_WITH_MSG_USERID .

MQXR2_USE_AGENT_BUFFER
Użyj buforu agenta.

Ta wartość wskazuje, że wszystkie dane, które mają zostać przekazane, znajdują się w pliku *AgentBuffer*, a nie w pliku *ExitBufferAddr*.

Wartością jest zero, co odpowiada wartości początkowej ustawionej podczas wywoływania wyjścia. Stała jest udostępniana na potrzeby dokumentacji.

MQXR2_USE_EXIT_BUFFER
Użyj buforu wyjścia.

Ta wartość wskazuje, że wszystkie dane, które mają zostać przekazane, znajdują się w pliku *ExitBufferAddr*, a nie w pliku *AgentBuffer*.

Należy ustawić tylko jedną z wartości MQXR2_USE_AGENT_BUFFER i MQXR2_USE_EXIT_BUFFER .

MQXR2_DEFAULT_CONTINUATION
Domyślna kontynuacja.

Kontynuacja z następnym wyjściem w łańcuchu zależy od odpowiedzi z ostatnio wywołanego wyjścia:

- Jeśli zostanie zwrócona wartość MQXCC_SUPPRESS_FUNCTION lub MQXCC_CLOSE_CHANNEL, nie zostaną wywołane żadne dalsze wyjścia w łańcuchu.
- W przeciwnym razie zostanie wywołane następne wyjście w łańcuchu.

MQXR2_CONTINUE_CHAIN
Przejdź do następnego wyjścia.

MQXR2_SUPPRESS_CHAIN
Pomiń pozostałe wyjścia w łańcuchu.

Jest to pole wejściowe/wyjściowe do wyjścia.

Informacja zwrotna (MQLONG)
To pole określa kod informacji zwrotnej.

To pole jest ustawione na wartość MQFB_NONE dla wejścia do procedury zewnętrznej.

Jeśli wyjście komunikatu kanału ustawia pole *ExitResponse* na wartość MQXCC_SUPPRESS_FUNCTION, pole *Feedback* określa kod sprzężenia zwrotnego, który identyfikuje przyczynę umieszczenia komunikatu w kolejce niedostarczonych komunikatów (niedostarczonych komunikatów), a także służy do wysyłania raportu o wyjątkach, jeśli taki komunikat został zażądany. W tym przypadku, jeśli pole *Feedback* ma wartość MQFB_NONE, używany jest następujący kod sprzężenia zwrotnego:

MQFB_STOPPED_BY_MSG_EXIT
Komunikat został zatrzymany przez wyjście komunikatu kanału.

Wartość zwracana w tym polu przez wyjścia zabezpieczeń kanału, wysyłania, odbierania i ponawiania komunikatów nie jest używana przez agent MCA.

Wartość zwracana w tym polu przez wyjścia automatycznej definicji nie jest używana, jeśli *ExitResponse* ma wartość MQXCC_OK, ale w przeciwnym razie jest używana dla parametru *AuxErrorDataInt1* w komunikacie zdarzenia.

Jest to pole wejściowe/wyjściowe z wyjścia.

MaxSegment(MQLONG)
To pole określa maksymalną długość w bajtach, która może być wysłana w pojedynczej transmisji.

Nie jest on używany przez wyjście automatycznej definicji. Jest to interesujące dla wyjścia wysyłania kanału, ponieważ to wyjście musi zapewnić, że nie zwiększy wielkości segmentu transmisji do wartości większej niż *MaxSegmentLength*. Długość obejmuje początkowe 8 bajtów, których wyjście nie może zmienić. Wartość ta jest negocjowana między funkcjami kanału IBM MQ podczas inicjowania kanału. Więcej informacji na temat długości segmentów zawiera sekcja Tworzenie programów obsługi wyjścia kanału .

Wartość w tym polu nie ma znaczenia, jeśli *ExitReason* to MQXR_INIT.

Jest to pole wejściowe do wyjścia.

Obszar ExitUser(MQBYTE16)
To pole określa obszar użytkownika programu zewnętrznego-pole dostępne do użycia przez program zewnętrzny.

Jest on inicjowany jako binarny zero przed pierwszym wywołaniem wyjścia (które ma parametr *ExitReason* ustawiony na wartość MQXR_INIT), a następnie wszystkie zmiany wprowadzone w tym polu przez wyjście są zachowywane między wywołaniami wyjścia.

Zdefiniowana jest następująca wartość:

BRAK MQXUA
Brak informacji o użytkowniku.

Wartością długości pola jest zero binarne.

W języku programowania C zdefiniowana jest również stała MQXUA_NONE_ARRAY. Ta stała ma taką samą wartość jak MQXUA_NONE, ale jest tablicą znaków, a nie łańcuchem.

Długość tego pola jest określona przez wartość MQ_EXIT_USER_AREA_LENGTH. Jest to pole wejściowe/wyjściowe do wyjścia.

ExitData (MQCHAR32)
To pole określa dane wyjścia.

To pole jest ustawiane na wejściu do procedury wyjścia w celu uzyskania informacji, które funkcje kanału IBM MQ odebrały z definicji kanału. Jeśli takie informacje nie są dostępne, pole to jest puste.

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH.

Jest to pole wejściowe do wyjścia.

Poniższe pola w tej strukturze nie są dostępne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

MsgRetry(MQLONG)-liczba
To pole określa, ile razy komunikat był ponawiany.

Przy pierwszym wywołaniu wyjścia dla konkretnego komunikatu pole to ma wartość zero (nie podjęto jeszcze próby ponowienia). Przy każdym kolejnym wywołaniu wyjścia dla tego komunikatu wartość jest zwiększana o jeden przez agent MCA.

Jest to pole wejściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli *ExitReason* to MQXR_INIT. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

Odstęp czasu MsgRetry(MQLONG)
To pole określa minimalny przedział czasu (w milisekundach), po którym podejmowana jest ponowna próba wykonania operacji umieszczania.

Przy pierwszym wywołaniu wyjścia dla konkretnego komunikatu to pole zawiera wartość atrybutu kanału *MsgRetryInterval* . Wyjście może pozostawić tę wartość bez zmian lub zmodyfikować ją, aby określić inny przedział czasu w milisekundach. Jeśli wyjście zwróci wartość MQXCC_OK w pliku *ExitResponse*, agent MCA czeka co najmniej przez ten czas przed ponowną próbą wykonania operacji MQOPEN lub MQPUT. Podany przedział czasu musi być równy zero lub większy.

Drugie i kolejne wywołanie wyjścia dla tego komunikatu. To pole zawiera wartość zwróconą przez poprzednie wywołanie wyjścia.

Jeśli wartość zwrócona w polu *MsgRetryInterval* jest mniejsza od zera lub większa od 999 999 999, a parametr *ExitResponse* ma wartość MQXCC_OK, agent MCA ignoruje pole *MsgRetryInterval* w produkcie MQCXP i czeka na interwał określony przez atrybut kanału *MsgRetryInterval* .

Jest to pole wejściowe/wyjściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli *ExitReason* to MQXR_INIT. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

MsgRetryPrzyczyna (MQLONG)
To pole określa kod przyczyny z poprzedniej próby umieszczenia komunikatu.

To pole zawiera kod przyczyny z poprzedniej próby umieszczenia komunikatu. Jest to jedna z wartości MQRC_ *.

Jest to pole wejściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli *ExitReason* to MQXR_INIT. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_2.

Następujące pola w tej strukturze nie są dostępne, jeśli *Version* jest mniejsze niż MQCXP_VERSION_3.

HeaderLength (MQLONG)
To pole określa długość informacji nagłówka.

To pole dotyczy tylko wyjścia komunikatu i wyjścia ponowienia komunikatu. Wartość jest długością struktur nagłówka routingu na początku danych komunikatu. Są to struktura MQXQH, MQMDE (nagłówek rozszerzenia opisu komunikatu) oraz (w przypadku komunikatu listy dystrybucyjnej) struktura MQDH i tablice rekordów MQOR i MQPMR, które są zgodne ze strukturą MQXQH.

Wyjście komunikatu może sprawdzić te informacje nagłówka i zmodyfikować je w razie potrzeby, ale dane zwracane przez wyjście muszą być nadal w poprawnym formacie. Wyjście nie może na przykład szyfrować ani kompresować danych nagłówka na końcu wysyłającym, nawet jeśli wyjście komunikatu na końcu odbierającym wprowadza zmiany kompensujące.

Jeśli wyjście komunikatu modyfikuje informacje nagłówka w taki sposób, aby zmienić jego długość (na przykład przez dodanie innego miejsca docelowego do komunikatu listy dystrybucyjnej), musi zmienić odpowiednio wartość *HeaderLength* przed zwróceniem.

Jest to pole wejściowe/wyjściowe do wyjścia. Wartość w tym polu nie ma znaczenia, jeśli *ExitReason* to MQXR_INIT. Pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

PartnerName (MQCHAR48)
To pole określa nazwę partnera.

Nazwa partnera, w następujący sposób:

- W przypadku kanałów SVRCONN jest to identyfikator zalogowanego użytkownika na kliencie.
- Dla wszystkich innych typów kanałów jest to nazwa menedżera kolejek partnera.

Po zainicjowaniu wyjścia to pole jest puste, ponieważ menedżer kolejek nie zna nazwy partnera, dopóki nie nastąpi początkowa negocjacja.

Jest to pole wejściowe do wyjścia. Pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

Poziom FAPLevel (MQLONG)
Negocjowany poziom formatów i protokołów.

Jest to pole wejściowe do wyjścia. Zmiany w tym polu powinny być wprowadzane tylko pod kierownictwem usługi IBM . Pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

CapabilityFlags (MQLONG)
Opcję możliwości można ustawić na wartość MQCF_NONE lub MQCF_DIST_LISTS.

Można ustawić jedną z następujących opcji możliwości:

MQCF_BRAK
Brak flag.

MQCF_DIST_LISTS
Obsługiwane listy dystrybucyjne.

Jest to pole wejściowe do wyjścia. Pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

ExitNumber (MQLONG)
To pole określa numer porządkowy wyjścia.

Numer porządkowy wyjścia w obrębie typu zdefiniowanego w pliku *ExitId*. Na przykład, jeśli wywoływane wyjście jest trzecim zdefiniowanym wyjściem komunikatu, to pole zawiera wartość 3. Jeśli typem wyjścia jest taki, dla którego nie można zdefiniować listy wyjść (na przykład wyjścia zabezpieczeń), to pole ma wartość 1.

Jest to pole wejściowe do wyjścia. Pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_3.

Następujące pola w tej strukturze nie są obecne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_5.

ExitSpace (MQLONG)
Pole to określa liczbę bajtów w buforze transmisji zarezerwowaną dla wyjścia.

To pole dotyczy tylko wyjścia wysyłania. Określa ilość miejsca w bajtach, które funkcje kanału IBM MQ zarezerwują w buforze transmisji dla wyjścia. To pole umożliwia programowi obsługi wyjścia dodanie do buforu transmisji niewielkiej ilości danych (zwykle nie przekraczającej kilkuset bajtów) do użycia przez uzupełniający program obsługi wyjścia odbierania na drugim końcu. Dane dodane przez wyjście wysyłania muszą zostać usunięte przez wyjście odbierania.

W systemie z/OSwartość zawsze wynosi zero.

Uwaga: Funkcja ta nie może być używana do wysyłania dużych ilości danych, ponieważ może to spowodować obniżenie wydajności, a nawet wstrzymanie działania kanału.

Po ustawieniu wartości *ExitSpace* wyjście jest gwarantowane, że w buforze transmisji będzie zawsze dostępna co najmniej ta liczba bajtów, która będzie używana przez wyjście. Jednak wyjście może używać mniej niż ilość zarezerwowana lub więcej niż ilość zarezerwowana, jeśli w buforze transmisji jest dostępne miejsce. Obszar wyjścia w buforze jest udostępniany po istniejących danych.

Parametr *ExitSpace* może zostać ustawiony przez wyjście tylko wtedy, gdy parametr *ExitReason* ma wartość MQXR_INIT; we wszystkich innych przypadkach wartość zwracana przez wyjście jest ignorowana. Na wejściu do wyjścia parametr *ExitSpace* ma wartość zero dla wywołania MQXR_INIT i jest wartością zwracaną przez wywołanie MQXR_INIT w innych przypadkach.

Jeśli wartość zwrócona przez wywołanie MQXR_INIT jest ujemna lub w buforze transmisji jest mniej niż 1024 bajty dostępne dla danych komunikatu po zarezerwowaniu żądanego obszaru wyjścia dla wszystkich wyjść wysyłania w łańcuchu, agent MCA wyprowadza komunikat o błędzie i zamyka kanał. Podobnie, jeśli podczas przesyłania danych wyjścia w łańcuchu wyjścia wysyłania przydzielają więcej przestrzeni użytkownika niż są zarezerwowane, tak że mniej niż 1024 bajty pozostają w buforze transmisji dla danych komunikatu, agent MCA wyprowadza komunikat o błędzie i zamyka kanał. Limit 1024 umożliwia przetwarzanie przepływów sterowania i administracyjnych kanału przez łańcuch wyjść wysyłania bez konieczności segmentacji przepływów.

Jest to pole wejściowe/wyjściowe do wyjścia, jeśli *ExitReason* to MQXR_INIT, a pole wejściowe we wszystkich innych przypadkach. To pole nie jest obecne, jeśli *Version* jest mniejsze niż MQCXP_VERSION_5.

Identyfikator SSLCertUser(MQCHAR12)
Pole to określa UserId powiązany ze zdalnym certyfikatem.

Pole to jest puste na wszystkich platformach z wyjątkiem systemu z/OS

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż wartość MQCXP_VERSION_6.

Długość SSLRemCertIssName(MQLONG)
To pole określa długość (w bajtach) pełnej nazwy wyróżniającej wystawcy zdalnego certyfikatu wskazywanego przez parametr SSLCertRemoteIssuerName.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż wartość MQCXP_VERSION_6. Wartość wynosi zero, jeśli nie jest to kanał TLS.

SSLRemCertIssNamePtr (PMQVOID)
Pole to określa adres pełnej nazwy wyróżniającej wystawcy certyfikatu zdalnego.

Jego wartością jest wskaźnik pusty, jeśli nie jest to kanał TLS.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż wartość MQCXP_VERSION_6.

Uwaga: Zachowanie wyjść zabezpieczeń kanału podczas określania nazwy wyróżniającej podmiotu i nazwy wyróżniającej wystawcy zostało zmienione z IBM WebSphere MQ 7.1. Więcej informacji na ten temat zawiera sekcja Programy obsługi wyjścia zabezpieczeń kanału.

SecurityParms (PMQCSP)
To pole określa adres struktury MQCSP używanej do określania referencji uwierzytelniających.

Wartością początkową tego pola jest wskaźnik pusty.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż wartość MQCXP_VERSION_6.

Wartość zwracana przez wyjście w tym polu musi być użyteczna w produkcie IBM MQ do czasu wywołania MQXR_TERM.

Kompresja CurHdr(MQLONG)
To pole określa, która technika jest obecnie używana do kompresowania danych nagłówka.

Jest on ustawiony na jedną z następujących wartości:

MQCOMPRESS_NONE
Dane nagłówka nie są kompresowane.

SYSTEM MQCOMPRESS_SYSTEM
Dane nagłówka są kompresowane.

Wartość ta może zostać zmieniona przez wyjście komunikatu kanału nadawczego na jedną z wynegocjowanych obsługiwanych wartości, do których dostęp można uzyskać z pola listy HdrCompw MQCD. Umożliwia to kompresowanie danych nagłówka, które mają być wybierane dla każdego komunikatu na podstawie treści komunikatu. Zmieniona wartość jest używana tylko dla bieżącego komunikatu. Kanał zostanie zakończony, jeśli atrybut zostanie zmieniony na nieobsługiwaną wartość. Wartość jest ignorowana, jeśli została zmieniona poza wyjściem komunikatu kanału nadawczego.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż wartość MQCXP_VERSION_6.

Kompresja CurMsg(MQLONG)
To pole określa, która technika jest obecnie używana do kompresowania danych komunikatu.

Jest on ustawiony na jedną z następujących wartości:

MQCOMPRESS_NONE
Dane nagłówka nie są kompresowane.

MQCOMPRESS_RLE
Kompresja danych komunikatu jest wykonywana przy użyciu kodowania grupowego.

MQCOMPRESS_ZLIBFAST,
Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowana jest szybka kompresja.

MQCOMPRESS_ZLIBHIGH
Kompresja danych komunikatu jest wykonywana przy użyciu techniki kompresji zlib. Preferowany jest wysoki poziom kompresji.

Wartość ta może zostać zmieniona przez wyjście komunikatu kanału nadawczego na jedną z wynegocjowanych obsługiwanych wartości, do których dostęp można uzyskać z pola listy MsgCompw

MQCD. Umożliwia to zastosowanie techniki kompresji danych komunikatu, która ma być określana dla każdego komunikatu na podstawie treści komunikatu. Zmieniona wartość jest używana tylko dla bieżącego komunikatu. Kanał zostanie zakończony, jeśli atrybut zostanie zmieniony na nieobsługiwaną wartość. Wartość jest ignorowana, jeśli została zmieniona poza wyjściem komunikatu kanału nadawczego.

Jest to pole wejściowe/wyjściowe do wyjścia. To pole nie jest wyświetlane, jeśli wartość *Version* jest mniejsza niż wartość MQCXP_VERSION_6.

Hconn (MQHCONN)
To pole określa uchwyt połączenia używany przez wyjście, jeśli ma ono wykonywać wywołania MQI w ramach wyjścia.

To pole nie dotyczy wyjść działających w kanałach połączeń klienta, w których zawiera wartość MQHC_UNUSABLE_HCONN (-1).

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_7.

SharingConversations (MQBOOL)
To pole określa, czy konwersacja jest jedyną konwersacją, która może być obecnie uruchomiona w tej instancji kanału, czy też w tej instancji kanału może być uruchomiona więcej niż jedna konwersacja.

Wskazuje również, czy istnieje ryzyko, że program obsługi wyjścia zostanie zmieniony przez inny program obsługi wyjścia działający w tym samym czasie.

To pole dotyczy tylko programów obsługi wyjścia działających w kanałach połączenia z klientem lub serwerem.

Jest on ustawiony na jedną z następujących wartości:

FAŁSZ
Instancja wyjścia jest jedyną instancją wyjścia, która może być obecnie uruchomiona w tej instancji kanału. Dzięki temu wyjście może bezpiecznie zaktualizować pola MQCD bez rywalizacji z innymi wyjściami działającymi w innych instancjach kanału. To, czy zmiany w polach MQCD są wykonywane przez kanał, jest definiowane przez tabelę pól MQCD w produkcie “Zmiana pól MQCD w wyjściu kanału” na stronie 1565.

PRAWDA
Instancja wyjścia nie jest jedyną instancją wyjścia, która może być obecnie uruchomiona w tej instancji kanału. Wszelkie zmiany wprowadzone w zmaterializowanej tabeli zapytania nie są wykonywane przez kanał, z wyjątkiem zmian wymienionych w tabeli pól zmaterializowanej tabeli zapytania w pliku “Zmiana pól MQCD w wyjściu kanału” na stronie 1565 dla przyczyn wyjścia innych niż MQXR_INIT. Jeśli to wyjście aktualizuje pola MQCD, upewnij się, że nie ma rywalizacji między innymi wyjściami działającymi w innych konwersacjach w tym samym czasie, udostępniając szeregowanie między wyjściami działającymi w tej instancji kanału.

Jest to pole wejściowe do wyjścia. Pole nie jest dostępne, jeśli wartość *Version* jest mniejsza niż MQCXP_VERSION_7.

MCAUserSource (MQLONG)
To pole określa źródło podanego identyfikatora użytkownika MCA.

Może on zawierać jedną z następujących wartości:

MQUSRC_MAPA
Identyfikator użytkownika jest określony w atrybucie MCAUSER.

MQUSRC_CHANNEL
Identyfikator użytkownika jest pobierany od partnera przychodzącego lub określony w polu MCAUSER zdefiniowanym w obiekcie kanału.

Jest to pole wejściowe do wyjścia. To pole nie jest wyświetlane, jeśli wersja jest niższa niż MQCXP_VERSION_8.

Punkty pEntry(PMQIEP)
To pole określa adres punktu wejścia interfejsu dla wywołania MQI lub DCI.

To pole nie jest dostępne, jeśli wartość w polu *Wersja* jest mniejsza niż MQCXP_VERSION_8.

RemoteProduct (MQCHAR4)
To pole określa nazwę produktu zdalnego.

To pole identyfikuje produkt zdalny klienta, na przykład C lub Java, wyświetlany w polu **RPRODUCT** komendy DISPLAY CHSATUS.

To pole nie jest wyświetlane, jeśli wartość w polu *Wersja* jest mniejsza niż MQCXP_VERSION_9.

RemoteVersion (MQCHAR8)
To pole określa nazwę wersji zdalnej.

To pole identyfikuje wersję bibliotek klienta, która jest wyświetlana w polu **RVERSION** komendy DISPLAY CHSTATUS.

To pole nie jest wyświetlane, jeśli wartość w polu *Wersja* jest mniejsza niż MQCXP_VERSION_9.

Deklaracja C
Ta deklaracja jest deklaracją języka C dla struktury MQCXP.

```
typedef struct tagMQCXP MQCXP;
struct tagMQCXP {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    ExitId;            /* Type of exit */
  MQLONG    ExitReason;        /* Reason for invoking exit */
  MQLONG    ExitResponse;      /* Response from exit */
  MQLONG    ExitResponse2;     /* Secondary response from exit */
  MQLONG    Feedback;          /* Feedback code */
  MQLONG    MaxSegmentLength;  /* Maximum segment length */
  MQBYTE16  ExitUserArea;      /* Exit user area */
  MQCHAR32  ExitData;          /* Exit data */
  MQLONG    MsgRetryCount;     /* Number of times the message has been
                                  retried */
  MQLONG    MsgRetryInterval;  /* Minimum interval in milliseconds after
                                  which the put operation should be
                                  retried */
  MQLONG    MsgRetryReason;    /* Reason code from previous attempt to
                                  put the message */
  MQLONG    HeaderLength;      /* Length of header information */
  MQCHAR48  PartnerName;       /* Partner Name */
  MQLONG    FAPLevel;          /* Negotiated Formats and Protocols
                                  level */
  MQLONG    CapabilityFlags;   /* Capability flags */
  MQLONG    ExitNumber;        /* Exit number */
  /* Ver:3 */
  /* Ver:4 */
  MQLONG    ExitSpace;         /* Number of bytes in transmission buffer
                                  reserved for exit to use */
  /* Ver:5 */
  MQCHAR12  SSLCertUserid;     /* User identifier associated
                                  with remote TLS certificate */
  MQLONG    SSLRemCertIssNameLength;   /* Length of
                                  distinguished name of issuer
                                  of remote TLS certificate */
  MQPTR     SSLRemCertIssNamePtr;      /* Address of
                                  distinguished name of issuer
                                  of remote TLS certificate */
  PMQVOID   SecurityParms;     /* Security parameters */
  MQLONG    CurHdrCompression; /* Header data compression
                                  used for current message */
  MQLONG    CurMsgCompression; /* Message data compression
                                  used for current message */
  /* Ver:6 */
  MQHCONN   Hconn;             /* Connection handle */
  MQBOOL    SharingConversations;      /* Multiple conversations
                                  possible on channel inst?  */
  /* Ver:7 */
  MQLONG    MCAUserSource;     /* Source of the provided MCA user ID */
  PMQIEP    pEntryPoints;      /* Address of the MQIEP structure */
  /* Ver:8 */
```

```
  MQCHAR4   RemoteProduct;     /* The identifier for the remote product */
  MQCHAR8   RemoteVersion;     /* The version of the remote product */
  /* Ver:9 */
};
```

Deklaracja języka COBOL

Ta deklaracja jest deklaracją języka COBOL dla struktury MQCXP.

```
**   MQCXP structure
  10 MQCXP.
**    Structure identifier
   15 MQCXP-STRUCID          PIC X(4).
**    Structure version number
   15 MQCXP-VERSION          PIC S9(9) BINARY.
**    Type of exit
   15 MQCXP-EXITID           PIC S9(9) BINARY.
**    Reason for invoking exit
   15 MQCXP-EXITREASON       PIC S9(9) BINARY.
**    Response from exit
   15 MQCXP-EXITRESPONSE     PIC S9(9) BINARY.
**    Secondary response from exit
   15 MQCXP-EXITRESPONSE2    PIC S9(9) BINARY.
**    Feedback code
   15 MQCXP-FEEDBACK         PIC S9(9) BINARY.
**    Maximum segment length
   15 MQCXP-MAXSEGMENTLENGTH PIC S9(9) BINARY.
**    Exit user area
   15 MQCXP-EXITUSERAREA     PIC X(16).
**    Exit data
   15 MQCXP-EXITDATA         PIC X(32).
**    Number of times the message has been retried
   15 MQCXP-MSGRETRYCOUNT    PIC S9(9) BINARY.
**    Minimum interval in milliseconds after which the put operation
**    should be retried
   15 MQCXP-MSGRETRYINTERVAL PIC S9(9) BINARY.
**    Reason code from previous attempt to put the message
   15 MQCXP-MSGRETRYREASON   PIC S9(9) BINARY.
**    Length of header information
   15 MQCXP-HEADERLENGTH     PIC S9(9) BINARY.
**    Partner Name
   15 MQCXP-PARTNERNAME      PIC X(48).
**    Negotiated Formats and Protocols level
   15 MQCXP-FAPLEVEL         PIC S9(9) BINARY.
**    Capability flags
   15 MQCXP-CAPABILITYFLAGS  PIC S9(9) BINARY.
**    Exit number
   15 MQCXP-EXITNUMBER       PIC S9(9) BINARY.
**    Number of bytes in transmission buffer reserved for exit to use
   15 MQCXP-EXITSPACE        PIC S9(9) BINARY.
**    User Id associated with remote certificate
   15  MQCXP-SSLCERTUSERID   PIC X(12).
** Length of distinguished name of issuer of remote TLS
** certificate
   15  MQCXP-SSLREMCERTISSNAMELENGTH PIC S9(9) BINARY.
** Address of distinguished name of issuer of remote TLS
** certificate
   15  MQCXP-SSLREMCERTISSNAMEPTR    POINTER.
** Security parameters
   15  MQCXP-SECURITYPARMS           PIC S9(18) BINARY.
** Header data compression used for current message
   15  MQCXP-CURHDRCOMPRESSION       PIC S9(9) BINARY.
** Message data compression used for current message
   15  MQCXP-CURMSGCOMPRESSION       PIC S9(9) BINARY.
** Connection handle
   15 MQCXP-HCONN                    PIC S9(9) BINARY.
** Multiple conversations possible on channel instance?
   15 MQCXP-SHARINGCONVERSATIONS     PIC S9(9) BINARY.
** Source of the provided MCA user ID
   15 MQCXP-MCAUSERSOURCE            PIC S9(9) BINARY.
** Identifier of the remote product
   15 MQCXP-RPRODUCT                 PIC X(4).
** Identifier of the remote version
   15 MQCXP-RVERSION                 PIC X(8).
```

Deklaracja RPG (ILE)

Ta deklaracja jest deklaracją języka RPG dla struktury MQCXP.

```
D*..1....:....2....:....3....:....4....:....5....:....6....:....7..
D* MQCXP Structure
D*
D* Structure identifier
D  CXSID                  1      4
D* Structure version number
D  CXVER                  5      8I 0
D* Type of exit
D  CXXID                  9     12I 0
D* Reason for invoking exit
D  CXREA                 13     16I 0
D* Response from exit
D  CXRES                 17     20I 0
D* Secondary response from exit
D  CXRE2                 21     24I 0
D* Feedback code
D  CXFB                  25     28I 0
D* Maximum segment length
D  CXMSL                 29     32I 0
D* Exit user area
D  CXUA                  33     48
D* Exit data
D  CXDAT                 49     80
D* Number of times the message has been retried
D  CXMRC                 81     84I 0
D* Minimum interval in milliseconds after which the put operation
D* should be retried
D  CXMRI                 85     88I 0
D* Reason code from previous attempt to put the message
D  CXMRR                 89     92I 0
D* Length of header information
D  CXHDL                 93     96I 0
D* Partner Name
D  CXPNM                 97    144
D* Negotiated Formats and Protocols level
D  CXFAP                145    148I 0
D* Capability flags
D  CXCAP                149    152I 0
D* Exit number
D  CXEXN                153    156I 0
D* Number of bytes in transmission buffer reserved for exit to use
D  CXHDL                157    160I 0
D* User identifier associated with remote TLS certificate
D  CXSSLCU              161    172
D* Length of distinguished name of issuer of remote TLS certificate
D  CXSRCINL             173    176I 0
D* Address of distinguished name of issuer of remote TLS certificate
D  CXSRCINP             177    192*
D* Security parameters
D  CXSECP               193    208*
D* Header data compression used for current message
D  CXCHC                209    212I 0
D* Message data compression used for current message
D  CXCMC                213    216I 0
D* Connection handle
D  CXHCONN               217    220I   0
D* Multiple conversations possible on channel instance?
D  CXSHARECONV           221    224I   0
D* Source of the provided MCA user ID
D  MCAUSERSOURCE         225    228I   0
D* Identifer of the remote product
D  CXRPRO                229    232I   0
D* Identifier of the remote version
D  CXRVER                233    240I   0
```

Deklaracja asemblera System/390

Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQCXP.

```
MQCXP                  DSECT
MQCXP_STRUCID          DS   CL4   Structure identifier
MQCXP_VERSION          DS   F     Structure version number
MQCXP_EXITID           DS   F     Type of exit
MQCXP_EXITREASON       DS   F     Reason for invoking exit
MQCXP_EXITRESPONSE     DS   F     Response from exit
MQCXP_EXITRESPONSE2    DS   F     Secondary response from exit
MQCXP_FEEDBACK         DS   F     Feedback code
```

```
MQCXP_MAXSEGMENTLENGTH  DS   F     Maximum segment length
MQCXP_EXITUSERAREA      DS   XL16  Exit user area
MQCXP_EXITDATA          DS   CL32  Exit data
MQCXP_MSGRETRYCOUNT     DS   F     Number of times the message has been
*                                  retried
MQCXP_MSGRETRYINTERVAL  DS   F     Minimum interval in milliseconds
*                                  after which the put operation should
*                                  be retried
MQCXP_MSGRETRYREASON    DS   F     Reason code from previous attempt to
*                                  put the message
MQCXP_HEADERLENGTH      DS   F     Length of header information
MQCXP_PARTNERNAME       DS   CL48  Partner Name
MQCXP_FAPLEVEL          DS   F     Negotiated Formats and Protocols
*                                  level
MQCXP_CAPABILITYFLAGS   DS   F     Capability flags
MQCXP_EXITNUMBER        DS   F     Exit number
MQCXP_EXITSPACE         DS   F     Number of bytes in transmission
*                                  buffer reserved for exit to use
MQCXP_SSLCERTUSERID     DS   CL12  User identifier associated with
*                                  remote TLS certificate
MQCXP_SSLREMCERTISSNAMELENGTH   DS   F     Length of distinguished name
*                                  of issuer of remote TLS certificate
MQCXP_SSLREMCERTISSNAMEPTR      DS   F     Address of distinguished name
*                                  of issuer of remote TLS certificate
MQCXP_SECURITYPARMS     DS   F    Address of security parameters
MQCXP_CURHDRCOMPRESSION         DS   F     Header data compression used for
*                                  current message
MQCXP_CURMSGCOMPRESSION         DS   F      Message data compression used for
*                                  current message
MQCXP_HCONN             DS   F Connection handle
MQCXP_SHARINGCONVERSATIONS      DS   F Multiple conversations possible on
*                                  channel inst?
MQCXP_MCAUSERSOURCE     DS   F    Source of the provided MCA user ID
MQCXP_RPRODUCT          DS   CL4  Identifer of the remote product
MQCXP_RVERSION          DS   CL8  Identifer of the remote version

MQCXP_LENGTH            EQU  *-MQCXP
                        ORG  MQCXP
MQCXP_AREA              DS   CL(MQCXP_LENGTH)
```

MQXWD-Wyjście z deskryptora oczekiwania

Struktura MQXWD jest parametrem wejścia/wyjścia w wywołaniu MQXWAIT.

Ta struktura jest obsługiwana tylko w systemie z/OS.

Odsyłacze pokrewne

“Pola” na stronie 1583
W tym temacie opisano wszystkie pola w strukturze MQXWD.

“Deklaracja C” na stronie 1584
Ta deklaracja jest deklaracją w języku C dla struktury MQXWD.

“Deklaracja asemblera System/390” na stronie 1584
Ta deklaracja jest deklaracją asemblera System/390 dla struktury MQXWD.

Pola

W tym temacie opisano wszystkie pola w strukturze MQXWD.

StrucId (MQCHAR4)
To pole określa identyfikator struktury.

Wartość musi być następująca:

Identyfikator struktury MQXWD_STRUC_ID
Identyfikator struktury deskryptora oczekiwania wyjścia.

Dla języka programowania C zdefiniowana jest również stała MQXWD_STRUC_ID_ARRAY; ta stała ma taką samą wartość jak MQXWD_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wartością początkową tego pola jest MQXWD_STRUC_ID.

Wersja (MQLONG)
To pole określa numer wersji struktury.

Wartość musi być następująca:

MQXWD_VERSION_1
Numer wersji struktury deskryptora oczekiwania na wyjście.

Wartością początkową tego pola jest MQXWD_VERSION_1.

Reserved1 (MQLONG)
To pole jest zarezerwowane. Jego wartość musi być równa zero.

Jest to pole wejściowe.

Reserved2 (MQLONG)
To pole jest zarezerwowane. Jego wartość musi być równa zero.

Jest to pole wejściowe.

Reserved3 (MQLONG)
To pole jest zarezerwowane. Jego wartość musi być równa zero.

Jest to pole wejściowe.

EBC (MQLONG)
To pole określa blok kontrolny zdarzenia, na który należy czekać.

To pole jest blokiem sterowania zdarzeniami (EBC), na który należy czekać. Musi być ustawiona na zero przed wywołaniem MQXWAIT, a po pomyślnym zakończeniu musi zawierać kod pocztowy.

To pole jest polem wejścia/wyjścia.

Deklaracja C
Ta deklaracja jest deklaracją w języku C dla struktury 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 */
};
```

Deklaracja asemblera System/390
Ta deklaracja jest deklaracją asemblera System/390 dla struktury 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)
```

Wywołania wyjścia obciążenia klastra i struktury danych

Ta sekcja zawiera informacje uzupełniające dotyczące wyjścia obciążenia klastra i struktur danych. Są to ogólne informacje o interfejsie programistycznym.

Wyjścia obciążenia klastra można pisać w następujących językach programowania:

- C
- Asembler System/390 (IBM MQ for z/OS)

Wywołanie jest opisane w:

- “MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia” na stronie 1585

Typy danych struktury używane przez wyjście są opisane w następujących sekcjach:

- “MQXCLWLN -nawigowanie po rekordach obciążenia klastra” na stronie 1587
- “MQWXP -Struktura parametru wyjścia obciążenia klastra” na stronie 1591
- “MQWDR-Struktura rekordu docelowego obciążenia klastra” na stronie 1599
- “MQWQR -Struktura rekordu kolejki obciążenia klastra” na stronie 1603
- “MQWCR -Struktura rekordu klastra obciążenia klastra” na stronie 1608
- z/OS Zachowanie asynchroniczne komend CLUSTER w systemie z/OS

W tej sekcji atrybuty menedżera kolejek i atrybuty kolejki są wyświetlane w całości. Równoważne nazwy używane w komendach MQSC zostały przedstawione poniżej. Szczegółowe informacje na temat komend MQSC zawiera sekcja Komendy MQSC.

Tabela 824. Atrybuty menedżera kolejek

Pełna nazwa	**Nazwa używana w MQSC**
ClusterWorkloadData	CLWLDATA
ClusterWorkloadExit	CLWLEXIT
ClusterWorkloadLength	CLWLLEN

Tabela 825. Kolejka - atrybuty

Pełna nazwa	**Nazwa używana w MQSC**
DefBind	DEFBIND
DefPersistence	DEFPSIST
DefPriority	DEFPRTY
InhibitPut	PUT
QDesc	DESCR

Zadania pokrewne

Zapisywanie i kompilowanie wyjść obciążenia klastra

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia

Wyjście obciążenia klastra jest wywoływane przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

Uwaga: Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQ_CLUSTER_WORKLOAD_EXIT . Zamiast tego nazwa wyjścia obciążenia klastra jest definiowana przez atrybut menedżera kolejek ClusterWorkload .

Wyjście MQ_CLUSTER_WORKLOAD_EXIT jest obsługiwane na wszystkich platformach.

Składnia

```
MQ_CLUSTER_WORKLOAD_EXIT (ExitParms)
```

Odsyłacze pokrewne

MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Wywołanie MQXCLWLN jest używane do przechodzenia między łańcuchami rekordów MQWDR, MQWQRi MQWCR zapisanymi w pamięci podręcznej klastra.

MQWXP -Struktura parametru wyjścia obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze parametru wyjścia obciążenia klastra MQWXP .

MQWDR-Struktura rekordu docelowego obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu obciążenia klastra MQWCR .

Parametry dla MQ_CLUSTER_WORKLOAD_EXIT

Opis parametrów w wywołaniu funkcji MQ_CLUSTER_WORKLOAD_EXIT .

***ExitParms* (MQWXP) -wejście/wyjście**
Blok parametru wyjścia.

- Program zewnętrzny ustawia informacje w systemie MQWXP , aby wskazać sposób zarządzania obciążeniem.

Odsyłacze pokrewne

Użycie notatek
Funkcja wykonywana przez wyjście obciążenia klastra jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami zdefiniowanymi w powiązanym bloku kontrolnym MQWXP.

Wywołania języka dla MQ_CLUSTER_WORKLOAD_EXIT
MQ_CLUSTER_WORKLOAD_EXIT obsługuje dwa języki: C i High Level Assembler.

Użycie notatek

Funkcja wykonywana przez wyjście obciążenia klastra jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami zdefiniowanymi w powiązanym bloku kontrolnym MQWXP.

Menedżer kolejek nie udostępnia żadnego punktu wejścia o nazwie MQ_CLUSTER_WORKLOAD_EXIT . Jednak dla nazwy MQ_CLUSTER_WORKLOAD_EXIT w języku programowania C podano nazwę typedef . Użyj parametru typedef do zadeklarowania wyjścia napisanego przez użytkownika, aby upewnić się, że parametry są poprawne.

Odsyłacze pokrewne

Parametry dla MQ_CLUSTER_WORKLOAD_EXIT
Opis parametrów w wywołaniu funkcji MQ_CLUSTER_WORKLOAD_EXIT .

Wywołania języka dla MQ_CLUSTER_WORKLOAD_EXIT
MQ_CLUSTER_WORKLOAD_EXIT obsługuje dwa języki: C i High Level Assembler.

Wywołania języka dla MQ_CLUSTER_WORKLOAD_EXIT

MQ_CLUSTER_WORKLOAD_EXIT obsługuje dwa języki: C i High Level Assembler.

Wywołanie C

```
MQ_CLUSTER_WORKLOAD_EXIT (&ExitParms);
```

Zastąp symbol *MQ_CLUSTER_WORKLOAD_EXIT* nazwą funkcji wyjścia obciążenia klastra.

Zadeklaruj parametry **MQ_CLUSTER_WORKLOAD_EXIT** w następujący sposób:

```
MQWXP ExitParms; /* Exit parameter block */
```

Wywołanie programu High Level Assembler

```
CALL EXITNAME,(EXITPARMS)
```

Zadeklaruj parametry w następujący sposób:

```
EXITPARMS      CMQWXPA       Exit parameter block
```

Odsyłacze pokrewne

Parametry dla MQ_CLUSTER_WORKLOAD_EXIT
Opis parametrów w wywołaniu funkcji MQ_CLUSTER_WORKLOAD_EXIT .

Użycie notatek
Funkcja wykonywana przez wyjście obciążenia klastra jest definiowana przez dostawcę wyjścia. Jednak wyjście musi być zgodne z regułami zdefiniowanymi w powiązanym bloku kontrolnym MQWXP.

MQXCLWLN -nawigowanie po rekordach obciążenia klastra

Wywołanie MQXCLWLN jest używane do przechodzenia między łańcuchami rekordów MQWDR, MQWQRi MQWCR zapisanymi w pamięci podręcznej klastra.

Pamięć podręczna klastra jest obszarem pamięci głównej używanym do przechowywania informacji związanych z klastrem.

Jeśli pamięć podręczna klastra jest statyczna, ma stałą wielkość. Jeśli zostanie ona ustawiona na dynamiczną, pamięć podręczna klastra może zostać rozszerzona zgodnie z wymaganiami.

Ustaw typ pamięci podręcznej klastra na STATIC lub DYNAMIC przy użyciu parametru systemowego lub makra.

- Multi Użyj parametru systemowego ClusterCacheType w systemie Wiele platform.
- z/OS Użyj parametru CLCACHE w makrze CSQ6SYSP w systemie z/OS.

Składnia

```
MQXCLWLN (ExitParms, CurrentRecord, NextOffset, NextRecord, Compcode, Reason)
```

Odsyłacze pokrewne

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia
Wyjście obciążenia klastra jest wywoływane przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

MQWXP -Struktura parametru wyjścia obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze parametru wyjścia obciążenia klastra MQWXP .

MQWDR-Struktura rekordu docelowego obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu obciążenia klastra MQWCR .

Parametry dla MQXCLWLN -nawigowanie po rekordach obciążenia klastra

Opis parametrów w wywołaniu funkcji MQXCLWLN .

ExitParms (MQWXP) -wejście/wyjście
Blok parametru wyjścia.

Ta struktura zawiera informacje dotyczące wywołania wyjścia. Wyjście ustawia informacje w tej strukturze, aby wskazać sposób zarządzania obciążeniem.

CurrentRecord (MQPTR) -wejście
Adres bieżącego rekordu.

Struktura ta zawiera informacje dotyczące adresu rekordu aktualnie sprawdzanego przez wyjście. Rekord musi być jednego z następujących typów:

- Rekord docelowy obciążenia klastra (MQWDR)
- Rekord kolejki obciążenia klastra (MQWQR)
- Rekord klastra obciążenia klastra (MQWCR)

NextOffset (MQLONG) -wejście
Przesunięcie następnego rekordu.

Ta struktura zawiera informacje dotyczące przesunięcia następnego rekordu lub struktury. *NextOffset* jest wartością odpowiedniego pola przesunięcia w bieżącym rekordzie i musi być jednym z następujących pól:

- PoleChannelDefOffset w pakietu MQWDR
- Pole ClusterRecOffset w komponencie MQWDR
- PoleClusterRecOffset w pakietu MQWQR
- Pole ClusterRecOffset w komponencie MQWCR

NextRecord (MQPTR) -wyjście
Adres następnego rekordu lub struktury.

Ta struktura zawiera informacje dotyczące adresu następnego rekordu lub struktury. Jeśli *CurrentRecord* jest adresem pakietu MQWDR, a *NextOffset* jest wartością pola ChannelDefPrzesunięcie , *NextRecord* jest adresem struktury definicji kanału (MQCD).

Jeśli nie ma następnego rekordu lub struktury, menedżer kolejek ustawia parametr *NextRecord* na wskaźnik pusty, a wywołanie zwraca kod zakończenia MQCC_WARNING i kod przyczyny MQRC_NO_RECORD_AVAILABLE.

CompCode (MQLONG) -wyjście
Kod zakończenia.

Kod zakończenia ma jedną z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_WARNING
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED
Wywołanie nie powiodło się.

Przyczyna (MQLONG) -wyjście
Kwalifikujący się kod przyczyny CompCode

Jeśli CompCode ma wartość MQCC_OK:

MQRC_NONE
(0, X'0000')
Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_NO_RECORD_AVAILABLE
(2359, X'0937')
Brak dostępnego rekordu. Wywołano wywołanie MQXCLWLN z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Bieżący rekord jest ostatnim rekordem w łańcuchu. Działanie naprawcze: Brak.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_CURRENT_RECORD_ERROR
(2357, X'0935')
Parametr **CurrentRecord** jest niepoprawny. Wywołano wywołanie MQXCLWLN z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Adres określony przez parametr **CurrentRecord** nie jest adresem poprawnego rekordu.

CurrentRecord musi być adresem rekordu docelowego, rekordu MQWDR, rekordu kolejki (MQWQR) lub rekordu klastra (MQWCR) rezydujący w pamięci podręcznej klastra. Działanie naprawcze: Upewnij się, że wyjście obciążenia klastra przekazuje adres poprawnego rekordu rezydującego w pamięci podręcznej klastra.

MQRC_ENVIRONMENT_ERROR
(2012, X'07DC')
Wywołanie nie jest poprawne w środowisku. Wywołano wywołanie MQXCLWLN , ale nie z wyjścia obciążenia klastra.

MQRC_NEXT_OFFSET_ERROR
(2358, X'0936')
Parametr **NextOffset** jest niepoprawny. Wywołano wywołanie MQXCLWLN z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Przesunięcie określone przez parametr **NextOffset** jest niepoprawne. **NextOffset** musi być wartością jednego z następujących pól:

- PoleChannelDefOffset w pakietu MQWDR
- Pole ClusterRecOffset w komponencie MQWDR
- PoleClusterRecOffset w pakietu MQWQR
- Pole ClusterRecOffset w komponencie MQWCR

Czynność naprawcza: upewnij się, że wartość określona dla parametru **NextOffset** jest wartością jednego z pól wymienionych wcześniej.

MQRC_NEXT_RECORD_ERROR
(2361, X'0939')
Parametr **NextRecord** jest niepoprawny.

MQRC_WXP_ERROR
(2356, X'0934')
Struktura parametru wyjścia obciążenia jest niepoprawna. Wywołano wywołanie MQXCLWLN z wyjścia obciążenia klastra w celu uzyskania adresu następnego rekordu w łańcuchu. Struktura parametru wyjścia obciążenia **ExitParms** jest niepoprawna z jednej z następujących przyczyn:

- Wskaźnik parametru jest niepoprawny. Nie zawsze możliwe jest wykrycie niepoprawnych wskaźników parametrów; jeśli nie zostaną wykryte, wystąpią nieprzewidywalne wyniki.
- Pole StrucId nie ma wartości MQWXP_STRUC_ID.
- Pole Wersja nie ma wartości MQWXP_VERSION_2.
- Pole Kontekst nie zawiera wartości przekazanej do wyjścia przez menedżer kolejek.

Czynność naprawcza: upewnij się, że parametr określony dla **ExitParms** jest strukturą MQWXP , która została przekazana do wyjścia podczas wywoływania wyjścia.

Odsyłacze pokrewne
Uwagi dotyczące użycia MQXCLWLN-nawigowanie w rekordach obciążenia klastra
Parametr MQXCLWLN umożliwia przechodzenie między rekordami grup, nawet jeśli pamięć podręczna jest statyczna.

Wywołania języka MQXCLWLN
MQXCLWLN obsługuje dwa języki: C i High Level Assembler.

Uwagi dotyczące użycia MQXCLWLN-nawigowanie w rekordach obciążenia klastra
Parametr MQXCLWLN umożliwia przechodzenie między rekordami grup, nawet jeśli pamięć podręczna jest statyczna.

Jeśli pamięć podręczna klastra jest dynamiczna, do przechodzenia między rekordami należy używać wywołania MQXCLWLN . Wyjście kończy się nieprawidłowo, jeśli do przechodzenia między rekordami używana jest prosta arytmetyka wskaźnika i przesunięcia.

Jeśli pamięć podręczna klastra jest statyczna, nie należy używać opcji MQXCLWLN do przechodzenia między rekordami. Zwykle należy używać parametru MQXCLWLN nawet wtedy, gdy pamięć podręczna jest statyczna. Następnie można zmienić pamięć podręczną klastra na dynamiczną bez konieczności zmiany wyjścia obciążenia.

Odsyłacze pokrewne
Parametry dla MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Opis parametrów w wywołaniu funkcji MQXCLWLN .

Wywołania języka MQXCLWLN
MQXCLWLN obsługuje dwa języki: C i High Level Assembler.

Wywołania języka MQXCLWLN
MQXCLWLN obsługuje dwa języki: C i High Level Assembler.

Wywołanie C

```
MQXCLWLN (&ExitParms, CurrentRecord, NextOffset, &NextRecord, &CompCode, &Reason) ;
```

Zadeklaruj parametry w następujący sposób:

```
Typedef struct tagMQXCLWLN {
MQWXP   ExitParms;      /* Exit parameter block */
MQPTR   CurrentRecord;  /* Address of current record*/
MQLONG  NextOffset;     /* Offset of next record */
MQPTR   NextRecord;     /* Address of next record or structure */
MQLONG  CompCode;       /* Completion code */
MQLONG  Reason;         /* Reason code qualifying CompCode */
```

Wywołanie programu High Level Assembler

```
CALL MQXCLWLN,(CLWLEXITPARMS,CURRENTRECORD,NEXTOFFSET,NEXTRECORD,COMPCODE,REASON)
```

Zadeklaruj parametry w następujący sposób:

```
CLWLEXITPARMS CMQWXPA, Cluster workload exit parameter block
CURRENTRECORD CMQWDRA, Current record
NEXTOFFSET    DS F     Next offset
NEXTRECORD    DS F     Next record
COMPCODE      DS F     Completion code
REASON        DS F     Reason code qualifying COMPCODE
```

Odsyłacze pokrewne
Parametry dla MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Opis parametrów w wywołaniu funkcji MQXCLWLN .

Uwagi dotyczące użycia MQXCLWLN-nawigowanie w rekordach obciążenia klastra
Parametr MQXCLWLN umożliwia przechodzenie między rekordami grup, nawet jeśli pamięć podręczna jest statyczna.

MQWXP -Struktura parametru wyjścia obciążenia klastra

Poniższa tabela zawiera podsumowanie pól w strukturze parametru wyjścia obciążenia klastra MQWXP .

Tabela 826. Pola w produkcie MQWXP		
Pole	**Opis**	**Strona**
StrucId	Identyfikator struktury	StrucId
Version	Numer wersji struktury	Wersja
ExitId	Typ wyjścia	ExitId
ExitReason	Przyczyna wywołania wyjścia	ExitReason
ExitResponse	Odpowiedź z wyjścia	ExitResponse
ExitResponse2	Dodatkowa odpowiedź z wyjścia	ExitResponse2
Feedback	Kod zwrotny	Opinia
Flags	Oznacza wartości. Te flagi bitowe są używane do wskazania informacji o umieszczanym komunikacie.	Flagi
ExitUserArea	Wyjdź z obszaru użytkownika	ObszarExitUser
ExitData	Dane wyjścia	ExitData
MsgDescPtr	Adres deskryptora komunikatu (MQMD)	MsgDescPtr
MsgBufferPtr	Adres buforu zawierającego niektóre lub wszystkie dane komunikatu	MsgBufferPtr
MsgBufferLength	Długość buforu zawierającego dane komunikatu	MsgBufferDługość
MsgLength	Długość kompletnego komunikatu	MsgLength
QName	Nazwa kolejki	Nazwa QName
QMgrName	Nazwa lokalnego menedżera kolejek	QMgrName
DestinationCount	Liczba możliwych miejsc przeznaczenia	DestinationCount
DestinationChosen	Wybrane miejsce docelowe	DestinationChosen
DestinationArrayPtr	Adres tablicy wskaźników do rekordów docelowych (MQWDR)	DestinationArrayPtr
QArrayPtr	Adres tablicy wskaźników do rekordów kolejki (MQWQR)	QArrayPtr
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWXP_VERSION_2.		
CacheContext	Informacje o kontekście	CacheContext
CacheType	Typ pamięci podręcznej klastra	CacheType
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWXP_VERSION_3.		
CLWLMRUChannels	Maksymalna liczba dozwolonych aktywnych wychodzących kanałów klastra	Kanały CLWLMRUChannel
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWXP_VERSION_4.		
pEntryPoints	Adres struktury MQIEP umożliwiający wykonywanie wywołań MQI i DCI	PunktypEntry

Struktura parametru wyjścia obciążenia klastra opisuje informacje przekazywane do wyjścia obciążenia klastra.

Struktura parametru wyjścia obciążenia klastra jest obsługiwana na wszystkich platformach

Dodatkowo dostępne są struktury MQWXP1, MQWXP2 i MQWXP3 dla kompatybilności wstecznej.

Odsyłacze pokrewne

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia
Wyjście obciążenia klastra jest wywoływane przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Wywołanie MQXCLWLN jest używane do przechodzenia między łańcuchami rekordów MQWDR, MQWQRi MQWCR zapisanymi w pamięci podręcznej klastra.

MQWDR-Struktura rekordu docelowego obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu obciążenia klastra MQWCR .

Pola w produkcie MQWXP -struktura parametru wyjścia obciążenia klastra

Opis pól pakietu MQWXP -struktura parametru wyjścia obciążenia klastra

StrucId (MQCHAR4)-dane wejściowe
Identyfikator struktury dla struktury parametru wyjścia obciążenia klastra.

- Wartość StrucId to MQWXP_STRUC_ID.
- Dla języka programowania C zdefiniowana jest również stała MQWXP_STRUC_ID_ARRAY . Ma taką samą wartość jak MQWXP_STRUC_ID. Jest to tablica znaków zamiast łańcucha.

Wersja (MQLONG)-wejście
Wskazuje numer wersji struktury. Wersja przyjmuje jedną z następujących wartości:

MQWXP_VERSION_1
Struktura parametru wyjścia obciążenia klastra Version-1 .

Produkt MQWXP_VERSION_1 jest obsługiwany we wszystkich środowiskach.

MQWXP_VERSION_2
Struktura parametru wyjścia obciążenia klastra Version-2 .

Produkt MQWXP_VERSION_2 jest obsługiwany w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

MQWXP_VERSION_3
Struktura parametru wyjścia obciążenia klastra Version-3 .

Produkt MQWXP_VERSION_3 jest obsługiwany w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

MQWXP_VERSION_4
Struktura parametru wyjścia obciążenia klastra Version-4 .

Produkt MQWXP_VERSION_4 jest obsługiwany w następujących środowiskach:

- AIX
- IBM i
- Linux
- Windows

MQWXP_CURRENT_VERSION
Bieżąca wersja struktury parametru wyjścia obciążenia klastra.

ExitId (MQLONG)-wejście
Wskazuje typ wywoływanego wyjścia. Jedynym obsługiwanym wyjściem jest wyjście obciążenia klastra.

- Parametr ExitId musi mieć wartość MQXT_CLUSTER_WORKLOAD_EXIT

ExitReason (MQLONG)-wejście
Wskazuje przyczynę wywołania wyjścia obciążenia klastra. ExitReason przyjmuje jedną z następujących wartości:

MQXR_INIT
Wskazuje, że wyjście jest wywoływane po raz pierwszy.

Uzyskaj i zainicjuj wszystkie zasoby, których może potrzebować wyjście, takie jak pamięć główna.

MQXR_TERM
Wskazuje, że wyjście zostanie zakończone.

Zwolnij wszystkie zasoby, które mogły zostać uzyskane przez wyjście od czasu jego zainicjowania, takie jak pamięć główna.

MQXR_CLWL_OPEN
Wywoływana przez MQOPEN.

MQXR_CLWL_PUT
Wywoływana przez MQPUT lub MQPUT1.

MQXR_CLWL_MOVE
Wywoływana przez agent MCA po zmianie stanu kanału.

MQXR_CLWL_REPOS
Wywoływana przez MQPUT lub MQPUT1 dla komunikatu PCF menedżera repozytorium.

MQXR_CLWL_REPOS_MOVE
Wywoływana przez agent MCA dla komunikatu PCF menedżera repozytorium, jeśli stan kanału uległ zmianie.

ExitResponse (MQLONG)-dane wyjściowe
Ustaw wartość ExitResponse , aby wskazać, czy przetwarzanie komunikatu jest kontynuowane. Musi to być jedna z następujących wartości:

MQXCC_OK
Kontynuuj normalne przetwarzanie komunikatu.

- DestinationChosen identyfikuje miejsce docelowe, do którego ma zostać wysłany komunikat.

MQXCC_SUPPRESS_FUNCTION
Należy przerwać przetwarzanie komunikatu.

- Działania podejmowane przez menedżer kolejek zależą od przyczyny wywołania wyjścia:

Tabela 827. Działania podejmowane przez menedżer kolejek	
ExitReason	**Wykonana czynność**
– MQXR_CLWL_OPEN – MQXR_CLWL_REPOS – MQXR_CLWL_PUT	Wywołanie MQOPEN, MQPUTlub MQPUT1 kończy się niepowodzeniem z kodem zakończenia MQCC_FAILED i kodem przyczyny MQRC_STOPPED_BY_CLUSTER_EXIT.
– MQXR_CLWL_MOVE – MQXR_CLWL_REPOS_MOVE	Komunikat jest umieszczany w kolejce niedostarczonych komunikatów.

MQXCC_SUPPRESS_EXIT
Kontynuuj przetwarzanie bieżącego komunikatu w normalny sposób. Nie wywołuj ponownie wyjścia, dopóki menedżer kolejek nie zostanie zamknięty.

Menedżer kolejek przetwarza kolejne komunikaty tak, jakby atrybut ClusterWorkloadExit menedżera kolejek jest pusty. DestinationChosen identyfikuje miejsce docelowe, do którego wysyłany jest bieżący komunikat.

Każda inna wartość
Przetwórz komunikat tak, jakby podano parametr MQXCC_SUPPRESS_FUNCTION .

ExitResponse2 (MQLONG)-wejście/wyjście
Ustaw wartość ExitResponse2 , aby udostępnić więcej informacji dla menedżera kolejek.

- MQXR2_STATIC_CACHE jest wartością domyślną i jest ustawiana na wejściu do wyjścia.
- Jeśli właściwość ExitReason ma wartość MQXR_INIT, wyjście może ustawić jedną z następujących wartości w polu ExitResponse2:

  MQXR2_STATIC_CACHE
  Wyjście wymaga statycznej pamięci podręcznej klastra.

  – Jeśli pamięć podręczna klastra jest statyczna, wyjście nie musi używać wywołania MQXCLWLN do poruszania się po łańcuchach rekordów w pamięci podręcznej klastra.

  – Jeśli pamięć podręczna klastra jest dynamiczna, wyjście nie może poprawnie przechodzić między rekordami w pamięci podręcznej.

  Uwaga: Menedżer kolejek przetwarza powrót z wywołania MQXR_INIT tak, jakby wyjście zwróciło wartość MQXCC_SUPPRESS_EXIT w polu ExitResponse .

  MQXR2_DYNAMIC_CACHE
  Wyjście może działać ze statyczną lub dynamiczną pamięcią podręczną.

  – Jeśli wyjście zwraca tę wartość, musi użyć wywołania MQXCLWLN , aby poruszać się po łańcuchach rekordów w pamięci podręcznej klastra.

Opinia (MQLONG)-dane wejściowe
Pole zastrzeżone. Wartość wynosi zero.

Flagi (MQLONG)-wejście
Wskazuje informacje o umieszczanym komunikacie.

- Wartość opcji Flags wynosi MQWXP_PUT_BY_CLUSTER_CHL. Komunikat pochodzi z kanału klastra, a nie lokalnie lub z kanału innego niż kanał klastra. Innymi słowy komunikat pochodzi z innego menedżera kolejek klastra.

Zarezerwowane (MQLONG)-wejście
Pole zastrzeżone. Wartość wynosi zero.

ObszarExitUser (MQBYTE16)-wejście/wyjście
Ustaw ObszarExitUser na komunikację między wywołaniami programu zewnętrznego.

- ExitUserObszar jest inicjowany do zera binarnego przed pierwszym wywołaniem wyjścia. Wszystkie zmiany wprowadzone w tym polu przez wyjście są zachowywane w wywołaniach

wyjścia, które występują między wywołaniem MQCONN i zgodnym wywołaniem MQDISC . Pole jest resetowane do zera binarnego, gdy wystąpi wywołanie MQDISC .

- Pierwsze wywołanie wyjścia jest wskazywane przez pole ExitReason o wartości MQXR_INIT.
- Zdefiniowane są następujące stałe:

  MQXUA_NONE -łańcuch
  MQXUA_NONE_ARRAY -tablica znaków
  Brak informacji o użytkowniku. Obie stałe są binarnymi zerami dla długości pola.

  MQ_EXIT_USER_AREA_LENGTH
  Długość obszaru ExitUser.

ExitData (MQCHAR32)-wejście
Wartość atrybutu menedżera kolejek ClusterWorkloadData . Jeśli dla tego atrybutu nie zdefiniowano żadnej wartości, pole jest puste.

- Długość danych ExitData jest określona przez parametr MQ_EXIT_DATA_LENGTH.

MsgDescPtr (PMQMD)-dane wejściowe
Adres kopii deskryptora komunikatu (MQMD) dla przetwarzanego komunikatu.

- Wszystkie zmiany deskryptora komunikatu wprowadzone przez wyjście są ignorowane przez menedżer kolejek.
- Jeśli parametr ExitReason ma jedną z następujących wartości MsgDescPtr jest ustawiony na wskaźnik pusty, a do wyjścia nie jest przekazywany żaden deskryptor komunikatu:
  - MQXR_INIT
  - MQXR_TERM
  - MQXR_CLWL_OPEN

MsgBufferPtr (PMQVOID)-wejście
Adres buforu zawierającego kopię pierwszych MsgBufferDługość bajtów danych komunikatu.

- Wszystkie zmiany wprowadzone w danych komunikatu przez wyjście są ignorowane przez menedżer kolejek.
- Żadne dane komunikatu nie są przekazywane do wyjścia, gdy:
  - MsgDescPtr jest wskaźnikiem pustym.
  - Komunikat nie zawiera danych.
  - Atrybut menedżera kolejek ClusterWorkloadLength ma wartość zero.

  W takich przypadkach parametr MsgBufferPtr jest wskaźnikiem pustym.

MsgBufferLength (MQLONG)-wejście
Długość buforu zawierającego dane komunikatu przekazane do wyjścia.

- Długość jest kontrolowana przez atrybut menedżera kolejek ClusterWorkloadLength .
- Długość może być mniejsza niż długość całego komunikatu, patrz sekcja MsgLength.

MsgLength (MQLONG)-dane wejściowe
Długość całego komunikatu przekazanego do wyjścia.

- MsgBufferDługość może być mniejsza niż długość całego komunikatu.
- MsgLength ma wartość zero, jeśli parametr ExitReason ma wartość MQXR_INIT, MQXR_TERMlub MQXR_CLWL_OPEN.

QName (MQCHAR48)-dane wejściowe
Nazwa kolejki docelowej. Kolejka jest kolejką klastra.

- Długość nazwy QName wynosi MQ_Q_NAME_LENGTH.

QMgrName (MQCHAR48)-wejście
Nazwa lokalnego menedżera kolejek, który wywołał wyjście obciążenia klastra.

- Długość nazwy QMgrName wynosi MQ_Q_MGR_NAME_LENGTH.

DestinationCount (MQLONG)-wejście
Liczba możliwych miejsc docelowych. Miejsca docelowe są instancjami kolejki docelowej i są opisane przez rekordy miejsca docelowego.

- Rekord docelowy jest strukturą MQWDR . Dla każdej możliwej trasy do każdej instancji kolejki istnieje jedna struktura.
- Struktury pakietu MQWDR są adresowane przez tablicę wskaźników, patrz DestinationArrayPtr.

DestinationChosen (MQLONG)-wejście/wyjście
Wybrane miejsce docelowe.

- Numer struktury MQWDR , która identyfikuje trasę i instancję kolejki, do której ma zostać wysłany komunikat.
- Wartość należy do zakresu od 1 do DestinationCount.
- Na wejściu do wyjścia opcja DestinationChosen wskazuje trasę i instancję kolejki wybraną przez menedżer kolejek. Wyjście może zaakceptować ten wybór lub wybrać inną instancję trasy i kolejki.
- Wartość ustawiona przez wyjście musi być z zakresu 1- DestinationCount. Jeśli zostanie zwrócona dowolna inna wartość, menedżer kolejek użyje wartości DestinationChosen na wejściu do wyjścia.

DestinationArrayPtr (PPMQWDR)-wejście
Adres tablicy wskaźników do rekordów docelowych (MQWDR).

- Istnieje DestinationCount rekordów docelowych.

QArrayPtr (PPMQWQR)-wejście
Adres tablicy wskaźników do rekordów kolejki (MQWQR).

- Jeśli rekordy kolejki są dostępne, istnieje ich liczba DestinationCount .
- Jeśli nie są dostępne żadne rekordy kolejki, QArrayPtr jest wskaźnikiem pustym.

  Uwaga: QArrayPtr może być wskaźnikiem pustym, nawet jeśli wartość DestinationCount jest większa od zera.

CacheContext (MQPTR): wersja 2-wejście
Pole CacheContext jest zarezerwowane do użytku przez menedżer kolejek. Wyjście nie może zmieniać wartości tego pola.

CacheType (MQLONG): wersja 2-wejście
Pamięć podręczna klastra ma jeden z następujących typów:

MQCLCT_STATIC
Pamięć podręczna jest statyczna.

- Wielkość pamięci podręcznej jest stała i nie może rosnąć w miarę działania menedżera kolejek.
- Nie ma potrzeby używania wywołania MQXCLWLN do przechodzenia między rekordami w pamięci podręcznej tego typu.

MQCLCT_DYNAMIC
Pamięć podręczna jest dynamiczna.

- Wielkość pamięci podręcznej może zostać zwiększona w celu uwzględnienia różnych informacji o klastrze.
- W celu nawigowania po rekordach w pamięci podręcznej tego typu należy użyć wywołania MQXCLWLN .

CLWLMRUChannels (MQLONG): wersja 3-wejście
Wskazuje maksymalną liczbę aktywnych wychodzących kanałów klastra, które mają być używane przez algorytm wyboru obciążenia klastra.

- CLWLMRUChannels ma wartość od 1 do 999 999 999.

pEntryPoints (PMQIEP): wersja 4
Adres struktury MQIEP, za pośrednictwem której można tworzyć wywołania MQI i DCI.

Odsyłacze pokrewne
Wartości początkowe i deklaracje języków dla pakietu MQWXP
Wartości początkowe oraz deklaracje języka C i High Level Assembler dla pakietu MQWXP -struktura parametrów wyjścia obciążenia klastra.

Wartości początkowe i deklaracje języków dla pakietu MQWXP
Wartości początkowe oraz deklaracje języka C i High Level Assembler dla pakietu MQWXP -struktura parametrów wyjścia obciążenia klastra.

Tabela 828. Pola w produkcie MQWXP

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQWXP_STRUC_ID	'WXP¬'
Version	MQWXP_VERSION_2	2
ExitId	Brak	0
ExitReason	MQXCC_OK	0
ExitResponse	Brak	0
ExitResponse2	Brak	0
Flags	Brak	0
ExitUserArea	{MQXUA_NONE_ARRAY}	0
ExitData	Brak	""
MsgDescPtr	Brak	NULL
MsgBufferPtr	Brak	NULL
MsgBufferLength	Brak	0
MsgBufferPtr	Brak	0
QName	Brak	""
QMgrName	Brak	""
DestinationCount	Brak	0
DestinationChosen	Brak	0
DestinationArrayPtr	Brak	NULL
QArrayPtr	Brak	NULL
CacheContext	Brak	NULL
CacheType	MQCLCT_DYNAMIC	1
CLWLMRUChannels	Brak	0
pEntryPoints	Brak	NULL

Tabela 828. Pola w produkcie MQWXP (kontynuacja)

Nazwa pola	Nazwa stałej	Wartość stałej

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C zmienna makra MQWXP_DEFAULT zawiera wartości domyślne. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQWDR MyWXP = {MQWXP_DEFAULT};
```

Deklaracja C

```
typedef struct tagMQWXP {
  MQCHAR4   StrucId;              /* Structure identifier */
  MQLONG    Version;              /* Structure version number */
  MQLONG    ExitId;               /* Type of exit */
  MQLONG    ExitReason;           /* Reason for invoking exit */
  MQLONG    ExitResponse;         /* Response from exit */
  MQLONG    ExitResponse2;        /* Reserved */
  MQLONG    Feedback;             /* Reserved */
  MQLONG    Flags;                /* Flags */
  MQBYTE16  ExitUserArea;         /* Exit user area */
  MQCHAR32  ExitData;             /* Exit data */
  PMQMD     MsgDescPtr;           /* Address of message descriptor */
  PMQVOID   MsgBufferPtr;         /* Address of buffer containing some
                                     or all of the message data */
  MQLONG    MsgBufferLength;      /* Length of buffer containing message
                                     data */
  MQLONG    MsgLength;            /* Length of complete message */
  MQCHAR48  QName;                /* Queue name */
  MQCHAR48  QMgrName;             /* Name of local queue manager */
  MQLONG    DestinationCount;     /* Number of possible destinations */
  MQLONG    DestinationChosen;    /* Destination chosen */
  PPMQWDR   DestinationArrayPtr;  /* Address of an array of pointers to
                                     destination records */
  PPMQWQR   QArrayPtr;            /* Address of an array of pointers to
                                     queue records */
  /* version 1 */
  MQPTR     CacheContext;         /* Context information */
  MQLONG    CacheType;            /* Type of cluster cache */
  /* version 2 */
  MQLONG    CLWLMRUChannnels;     /* Maximum number of most recently
                                     used cluster channels */
  /* version 3 */
  PMQIEP    pEntryPoints;         /* Address of the MQIEP structure */
  /* version 4 */
};
```

High Level Assembler

```
MQWXP                           DSECT
MQWXP_STRUCID                   DS   CL4      Structure identifier
MQWXP_VERSION                   DS   F        Structure version number
MQWXP_EXITID                    DS   F        Type of exit
MQWXP_EXITREASON                DS   F        Reason for invoking exit
MQWXP_EXITRESPONSE              DS   F        Response from exit
MQWXP_EXITRESPONSE2             DS   F        Reserved
MQWXP_FEEDBACK                  DS   F        Reserved
MQWXP_RESERVED                  DS   F        Reserved
MQWXP_EXITUSERAREA              DS   XL16     Exit user area
MQWXP_EXITDATA                  DS   CL32     Exit data
MQWXP_MSGDESCPTR                DS   F        Address of message
*                                             descriptor
MQWXP_MSGBUFFERPTR              DS   F        Address of buffer containing
*                                             some or all of the message
*                                             data
MQWXP_MSGBUFFERLENGTH           DS   F        Length of buffer containing
*                                             message data
```

```
MQWXP_MSGLENGTH                 DS   F        Length of complete message
MQWXP_QNAME                     DS   CL48     Queue name
MQWXP_QMGRNAME                  DS   CL48     Name of local queue manager
MQWXP_DESTINATIONCOUNT          DS   F        Number of possible
*                                            destinations
MQWXP_DESTINATIONCHOSEN         DS   F        Destination chosen
MQWXP_DESTINATIONARRAYPTR       DS   F        Address of an array of
*                                            pointers to destination
*                                            records
MQWXP_QARRAYPTR                 DS   F        Address of an array of
*                                            pointers to queue records
MQWXP_CACHECONTEXT              DS   F        Context information
MQWXP_CACHETYPE                 DS   F        Type of cluster cache
MQWXP_CLWLMRUCHANNELS           DS   F        Number of most recently used
*                                            channels for workload balancing

MQWXP_LENGTH                    EQU  *-MQWXP  Length of structure
                                ORG  MQWXP
MQWXP_AREA                      DS   CL(MQWXP_LENGTH)
```

Odsyłacze pokrewne

Pola w produkcie MQWXP -struktura parametru wyjścia obciążenia klastra
Opis pól pakietu MQWXP -struktura parametru wyjścia obciążenia klastra

MQWDR-Struktura rekordu docelowego obciążenia klastra

Poniższa tabela zawiera podsumowanie pól w strukturze rekordu docelowego obciążenia klastra MQWDR .

Tabela 829. Pola w narzędziu MQWDR

Pole	Opis	Strona
StrucId	Identyfikator struktury	StrucId
Version	Numer wersji struktury	Wersja
StrucLength	Długość struktury produktu MQWDR .	StrucLength
QMgrFlags	Flagi menedżera kolejek	QMgrFlags
QMgrIdentifier	Identyfikator menedżera kolejek	QMgrIdentifier
QMgrName	Nazwa menedżera kolejek	QMgrName
ClusterRecOffset	Przesunięcie logiczne pierwszego rekordu klastra (MQWCR)	ClusterRecprzesunięcie
ChannelState	Stan kanału	ChannelState
ChannelDefOffset	Przesunięcie logiczne struktury definicji kanału (MQCD)	ChannelDefPrzesunięcie
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWDR_VERSION_2.		
DestSeqNumber	Docelowy numer kolejny kanału	DestSeqNumer
DestSeqFactor	Docelowy współczynnik sekwencji kanału dla ważenia	DestSeqWspółczynnik

Struktura rekordu docelowego obciążenia klastra zawiera informacje dotyczące jednego z możliwych miejsc docelowych dla komunikatu. Dla każdej instancji kolejki docelowej istnieje jedna struktura rekordu docelowego obciążenia klastra.

Struktura rekordu docelowego obciążenia klastra jest obsługiwana we wszystkich środowiskach.

Dodatkowo w celu zapewnienia kompatybilności wstecznej dostępne są struktury MQWDR1 i MQWDR2 .

Odsyłacze pokrewne

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia
Wyjście obciążenia klastra jest wywoływane przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Wywołanie MQXCLWLN jest używane do przechodzenia między łańcuchami rekordów MQWDR, MQWQRi MQWCR zapisanymi w pamięci podręcznej klastra.

MQWXP -Struktura parametru wyjścia obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze parametru wyjścia obciążenia klastra MQWXP .

MQWQR -Struktura rekordu kolejki obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

MQWCR -Struktura rekordu klastra obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu obciążenia klastra MQWCR .

Pola w strukturze rekordu docelowego obciążenia produktu MQWDR-Cluster

Opis parametrów w strukturze rekordu docelowego obciążenia klastra MQWDR .

StrucId (MQCHAR4) -wejście
Identyfikator struktury dla struktury rekordu docelowego obciążenia klastra.

- Wartość StrucId to MQWDR_STRUC_ID.
- Dla języka programowania C zdefiniowana jest również stała MQWDR_STRUC_ID_ARRAY . Ma taką samą wartość jak MQWDR_STRUC_ID. Jest to tablica znaków zamiast łańcucha.

Wersja (MQLONG) -wejście
Numer wersji struktury. Wersja przyjmuje jedną z następujących wartości:

MQWDR_VERSION_1
Version-1 rekord docelowy obciążenia klastra.

MQWDR_VERSION_2
Version-2 rekord miejsca docelowego obciążenia klastra.

MQWDR_CURRENT_VERSION
Bieżąca wersja rekordu docelowego obciążenia klastra.

StrucLength (MQLONG) -wejście
Długość struktury MQWDR . StrucLength przyjmuje jedną z następujących wartości:

MQWDR_LENGTH_1
Długość rekordu docelowego obciążenia klastra w wersji version-1 .

MQWDR_LENGTH_2
Długość rekordu docelowego obciążenia klastra w wersji version-2 .

MQWDR_CURRENT_LENGTH
Długość bieżącej wersji rekordu docelowego obciążenia klastra.

QMgrFlags (MQLONG) -wejście
Flagi menedżera kolejek wskazujące właściwości menedżera kolejek udostępniającego instancję kolejki docelowej opisaną przez strukturę MQWDR . Zdefiniowane są następujące opcje:

MQQMF_REPOSITORY_Q_MGR
Miejsce docelowe jest menedżerem kolejek repozytorium pełnego.

MQQMF_CLUSSDR_USER_DEFINED
Kanał nadawczy klastra został zdefiniowany ręcznie.

MQQMF_CLUSSDR_AUTO_DEFINED
Kanał nadawczy klastra został zdefiniowany automatycznie.

MQQMF_AVAILABLE
Docelowy menedżer kolejek jest dostępny na potrzeby odbierania komunikatów.

Inne wartości
Inne flagi w tym polu mogą być ustawiane przez menedżer kolejek na potrzeby wewnętrzne.

QMgrIdentifier (MQCHAR48) -wejście
Identyfikator menedżera kolejek jest unikalnym identyfikatorem menedżera kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWDR .

- Identyfikator jest generowany przez menedżer kolejek.
- Długość identyfikatora QMgrIdentifier wynosi MQ_Q_MGR_IDENTIFIER_LENGTH.

QMgrName (MQCHAR48) -wejście
Nazwa menedżera kolejek udostępniającego instancję kolejki docelowej opisaną w strukturze MQWDR .

- QMgrName może być nazwą lokalnego menedżera kolejek oraz innego menedżera kolejek w klastrze.
- Długość nazwy QMgrName wynosi MQ_Q_MGR_NAME_LENGTH.

ClusterRecPrzesunięcie (MQLONG) -wejście
Przesunięcie logiczne pierwszej struktury MQWCR , która należy do struktury MQWDR .

- W przypadku statycznych pamięci podręcznych przesunięcie ClusterRec to przesunięcie pierwszej struktury MQWCR , która należy do struktury MQWDR .
- Przesunięcie jest mierzone w bajtach od początku struktury MQWDR .
- Nie należy używać przesunięcia logicznego dla arytmetyki wskaźników z dynamicznymi pamięciami podręcznymi. Aby uzyskać adres następnego rekordu, należy użyć wywołania MQXCLWLN .

ChannelState (MQLONG) -wejście
Stan kanału, który łączy lokalny menedżer kolejek z menedżerem kolejek identyfikowanym przez strukturę MQWDR . Dozwolone są następujące wartości:

MQCHS_BINDING
Kanał negocjuje z partnerem.

MQCHS_INACTIVE
Kanał nie jest aktywny.

MQCHS_INITIALIZING
Kanał jest inicjowany.

MQCHS_PAUSED
Kanał został wstrzymany.

MQCHS_REQUESTING
Kanał requestera żąda połączenia.

MQCHS_RETRYING
Kanał próbuje ponownie nawiązać połączenie.

MQCHS_RUNNING
Kanał przesyła lub oczekuje na komunikaty.

MQCHS_STARTING
Kanał oczekuje na aktywowanie.

MQCHS_STOPPING
Kanał jest zatrzymywany.

MQCHS_STOPPED
Kanał został zatrzymany.

ChannelDefChannelDef (MQLONG) -wejście
Przesunięcie logiczne definicji kanału (MQCD) dla kanału, który łączy lokalny menedżer kolejek z menedżerem kolejek identyfikowanym przez strukturę MQWDR .

- ChannelDefprzesunięcie jest podobne do ClusterRecprzesunięcie
- Przesunięcie logiczne nie może być używane w arytmetycznym wskaźniku. Aby uzyskać adres następnego rekordu, należy użyć wywołania MQXCLWLN .

DestSeqFactor (MQLONG) -wejście
Docelowy współczynnik sekwencji, który umożliwia wybór kanału na podstawie wagi.

- DestSeqWspółczynnik jest używany przed zmianą przez menedżera kolejek.
- Menedżer obciążenia zwiększa DestSeqDestSeq w taki sposób, aby komunikaty były rozdzielane w dół kanałów zgodnie z ich wagą.

DestSeqNumer (MQLONG) -wejście
Wartość miejsca docelowego kanału klastra przed zmianą menedżera kolejek.

- Menedżer obciążenia zwiększa wartość DestSeqNumber za każdym razem, gdy komunikat jest umieszczany w tym kanale.
- Wyjścia obciążenia mogą używać DestSeqNumber , aby zdecydować, który kanał ma umieścić komunikat w dół.

Odsyłacze pokrewne
Początkowe wartości i deklaracje języka dla pakietu MQWDR
Wartości początkowe oraz deklaracje języka C i High Level Assembler dla rekordu docelowego obciążenia klastra MQWDR .

Początkowe wartości i deklaracje języka dla pakietu MQWDR

Wartości początkowe oraz deklaracje języka C i High Level Assembler dla rekordu docelowego obciążenia klastra MQWDR .

Tabela 830. Pola w narzędziu MQWDR

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQWDR_STRUC_ID	'WDR¬'
Version	MQWDR_VERSION_1	1
StrucLength	MQWDR_CURRENT_LENGTH [3]	136
QMgrFlags	MQWDR_NONE	0
QMgrIdentifier	Brak	""
QMgrName	Brak	""
ClusterRecOffset	Brak	0
ChannelState	Brak	0
ChannelDefOffset	Brak	0
DestSeqNumber	Brak	0
DestSeqFactor	Brak	0

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C zmienna makra MQWDR_DEFAULT zawiera wartości domyślne. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQWDR MyWDR = {MQWDR_DEFAULT};
```

3. Wartości początkowe celowo ustawiły długość struktury na długość bieżącej wersji, a nie na wersję 1 struktury.

High Level Assembler

```
MQWDR                           DSECT
MQWDR_STRUCID                   DS   CL4      Structure identifier
MQWDR_VERSION                   DS   F        Structure version number
MQWDR_STRUCLENGTH               DS   F        Length of MQWDR structure
MQWDR_QMGRFLAGS                 DS   F        Queue manager flags
MQWDR_QMGRIDENTIFIER            DS   CL48     Queue manager identifier
MQWDR_QMGRNAME                  DS   CL48     Queue manager name
MQWDR_CLUSTERRECOFFSET          DS   F        Offset of first cluster
```

```
*                                       record
MQWDR_CHANNELSTATE              DS   F  Channel state
MQWDR_CHANNELDEFOFFSET          DS   F  Offset of channel definition
*                                       structure
MQWDR_LENGTH                    EQU  *-MQWDR  Length of structure
                                ORG  MQWDR
MQWDR_AREA                      DS   CL(MQWDR_LENGTH)
```

Deklaracja C

```
typedef struct tagMQWDR {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    StrucLength;       /* Length of MQWDR structure */
  MQLONG    QMgrFlags;         /* Queue managerflags */
  MQCHAR48  QMgrIdentifier;    /* Queue manageridentifier */
  MQCHAR48  QMgrName;          /* Queue manager name */
  MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
  MQLONG    ChannelState;      /* Channel state */
  MQLONG    ChannelDefOffset;  /* Offset of channel definition structure */
  /* Ver:1 */
  MQLONG    DestSeqNumber;     /* Cluster channel destination sequence number */
  MQINT64   DestSeqFactor;     /* Cluster channel factor sequence number */
  /* Ver:2 */
};
```

Odsyłacze pokrewne

Pola w strukturze rekordu docelowego obciążenia produktu MQWDR-Cluster
Opis parametrów w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra

Poniższa tabela zawiera podsumowanie pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

Tabela 831. Pola w MQWQR		
Pole	**Opis**	**Strona**
StrucId	Identyfikator struktury	StrucId
Version	Numer wersji struktury	Wersja
StrucLength	Długość struktury MQWQR	StrucLength
QFlags	Flagi kolejki	Opcje QFlags
QName	Nazwa kolejki	Nazwa QName
QMgrIdentifier	Identyfikator menedżera kolejek	QMgrIdentifier
ClusterRecOffset	Przesunięcie pierwszego rekordu klastra (MQWCR)	ClusterRecprzesunięcie
QType	Typ kolejki	QTYPE
QDesc	Opis kolejki	QDesc (Opis kolejek)
DefBind	Domyślne łączenie	DefBind
DefPersistence	Domyślna trwałość komunikatu	DefPersistence
DefPriority	Domyślny priorytet komunikatu	DefPriority
InhibitPut	Określa, czy operacje umieszczania w kolejce są dozwolone	InhibitPut
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWQR_VERSION_2.		

Tabela 831. Pola w MQWQR (kontynuacja)		
Pole	**Opis**	**Strona**
CWLQueuePriority	Wartość z zakresu od 0 do 9 reprezentująca priorytet kolejki	CLWLQueuePriority
CLWLQueueRank	Wartość z zakresu od 0 do 9 reprezentująca klasyfikację kolejki	CLWLQueueRank
Uwaga: Pozostałe pola są ignorowane, jeśli wersja jest mniejsza niż MQWQR_VERSION_3.		
DefPutResponse	Operacja put - domyślna odp.	OdpowiedźDefPut

Struktura rekordu kolejki obciążenia klastra zawiera informacje dotyczące jednego z możliwych miejsc docelowych dla komunikatu. Dla każdej instancji kolejki docelowej istnieje jedna struktura rekordu kolejki obciążenia klastra.

Struktura rekordu kolejki obciążenia klastra jest obsługiwana we wszystkich środowiskach.

Dodatkowo dostępne są struktury MQWQR1 i MQWQR2 , aby zapewnić kompatybilność wsteczną.

Odsyłacze pokrewne

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia
Wyjście obciążenia klastra jest wywoływane przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Wywołanie MQXCLWLN jest używane do przechodzenia między łańcuchami rekordów MQWDR, MQWQRi MQWCR zapisanymi w pamięci podręcznej klastra.

MQWXP -Struktura parametru wyjścia obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze parametru wyjścia obciążenia klastra MQWXP .

MQWDR-Struktura rekordu docelowego obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWCR -Struktura rekordu klastra obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu obciążenia klastra MQWCR .

Pola w produkcie MQWQR -Struktura rekordu kolejki obciążenia klastra

Opis pól pakietu MQWQR -Struktura rekordu kolejki obciążenia klastra.

StrucId (MQCHAR4) -wejście
Identyfikator struktury dla struktury rekordu kolejki obciążenia klastra.

- Wartość StrucId to MQWQR_STRUC_ID.
- Dla języka programowania C zdefiniowana jest również stała MQWQR_STRUC_ID_ARRAY . Ma taką samą wartość jak MQWQR_STRUC_ID. Jest to tablica znaków zamiast łańcucha.

Wersja (MQLONG) -wejście
Numer wersji struktury. Wersja przyjmuje jedną z następujących wartości:

MQWQR_VERSION_1
Version-1 rekord kolejki obciążenia klastra.

MQWQR_VERSION_2
Version-2 rekord kolejki obciążenia klastra.

MQWQR_VERSION_3
Rekord kolejki obciążenia klastra Version-3 .

MQWQR_CURRENT_VERSION
Bieżąca wersja rekordu kolejki obciążenia klastra.

StrucLength (MQLONG) -wejście
Długość struktury pakietu MQWQR . StrucLength przyjmuje jedną z następujących wartości:

MQWQR_LENGTH_1
Długość rekordu kolejki obciążenia klastra w wersji version-1 .

MQWQR_LENGTH_2
Długość rekordu kolejki obciążenia klastra w wersji version-2 .

MQWQR_LENGTH_3
Długość rekordu kolejki obciążenia klastra w wersji version-3 .

MQWQR_CURRENT_LENGTH
Długość bieżącej wersji rekordu kolejki obciążenia klastra.

QFlags (MQLONG) -wejście
Flagi kolejki wskazują właściwości kolejki. Zdefiniowane są następujące opcje:

MQQF_LOCAL_Q
Miejsce docelowe jest kolejką lokalną.

MQQF_CLWL_USEQ_ANY
Zezwala na użycie kolejek lokalnych i zdalnych w danych put.

MQQF_CLWL_USEQ_LOCAL
Zezwalaj tylko na operacje umieszczania w kolejce lokalnej.

Inne wartości
Inne flagi w tym polu mogą być ustawiane przez menedżer kolejek na potrzeby wewnętrzne.

Nazwa QName (MQCHAR48) -wejście
Nazwa kolejki, która jest jednym z możliwych miejsc docelowych komunikatu.

- Długość nazwy QName wynosi MQ_Q_NAME_LENGTH.

QMgrIdentifier (MQCHAR48) -wejście
Identyfikator menedżera kolejek jest unikalnym identyfikatorem menedżera kolejek, który udostępnia instancję kolejki opisaną przez strukturę MQWQR .

- Identyfikator jest generowany przez menedżer kolejek.
- Długość identyfikatora QMgrIdentifier wynosi MQ_Q_MGR_IDENTIFIER_LENGTH.

ClusterRecPrzesunięcie (MQLONG) -wejście
Przesunięcie logiczne pierwszej struktury MQWCR , która należy do struktury MQWQR .

- W przypadku statycznych pamięci podręcznych przesunięcie ClusterRec to przesunięcie pierwszej struktury MQWCR , która należy do struktury MQWQR .
- Przesunięcie jest mierzone w bajtach od początku struktury MQWQR .
- Nie należy używać przesunięcia logicznego dla arytmetyki wskaźników z dynamicznymi pamięciami podręcznymi. Aby uzyskać adres następnego rekordu, należy użyć wywołania MQXCLWLN .

QType (MQLONG) -wejście
Typ kolejki docelowej. Dozwolone są następujące wartości:

MQCQT_LOCAL_Q
Kolejka lokalna.

MQCQT_ALIAS_Q
Kolejka aliasowa.

MQCQT_REMOTE_ Q
Kolejka zdalna.

MQCQT_Q_MGR_ALIAS
Alias menedżera kolejek.

QDesc (MQCHAR64) -wejście
Atrybut kolejki opisu kolejki zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR .

- Długość QDesc wynosi MQ_Q_DESC_LENGTH.

DefBind (MQLONG) -wejście
Domyślny atrybut powiązania kolejki zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR . W przypadku korzystania z grup z klastrami należy określić parametr MQBND_BIND_ON_OPEN lub MQBND_BIND_ON_GROUP . Możliwe są następujące wartości:

MQBND_BIND_ON_OPEN
Powiązanie ustalone przez wywołanie MQOPEN .

MQBND_BIND_NOT_FIXED
Powiązanie nie jest stałe.

MQBND_BIND_ON_GROUP
Umożliwia aplikacji żądanie przydzielenia grupy komunikatów do tej samej instancji docelowej.

DefPersistence (MQLONG) -wejście
Domyślny atrybut kolejki trwałości komunikatów zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR . Dozwolone są następujące wartości:

MQPER_PERSISTENT
Komunikat jest trwały.

MQPER_NOT_PERSISTENT
Komunikat nie jest trwały.

DefPriority (MQLONG) -wejście

Domyślny atrybut kolejki priorytetu komunikatów zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR . Zakres priorytetu wynosi od 0 do MaxPriority.

- 0 oznacza najniższy priorytet.
- MaxPriority to atrybut menedżera kolejek, który udostępnia tę instancję kolejki docelowej.

InhibitPut (MQLONG) -wejście
Atrybut kolejki bez możliwości umieszczania zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR . Dozwolone są następujące wartości:

MQQA_PUT_INHIBITED
Operacje umieszczania są zablokowane.

MQQA_PUT_ALLOWED
Operacje umieszczania (put) są dozwolone.

CLWLQueuePriority (MQLONG) -wejście
Atrybut priorytetu kolejki obciążenia klastra zdefiniowany w menedżerze kolejek, który obsługuje instancję kolejki docelowej opisanej w strukturze MQWQR .

CLWLQueueRank (MQLONG) -wejście
Pozycja kolejki obciążenia klastra zdefiniowana w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR .

OdpowiedźDefPut (MQLONG) -wejście
Atrybut domyślnej kolejki odpowiedzi umieszczania zdefiniowany w menedżerze kolejek, który udostępnia instancję kolejki docelowej opisanej w strukturze MQWQR . Dozwolone są następujące wartości:

MQPRT_SYNC_RESPONSE
Synchroniczna odpowiedź na wywołania MQPUT lub MQPUT1 .

MQPRT_ASYNC_RESPONSE
Asynchroniczna odpowiedź na wywołania MQPUT lub MQPUT1 .

Odsyłacze pokrewne

Wartości początkowe i deklaracje językowe dla MQWQR
Wartości początkowe oraz deklaracje języka C i High Level Assembler dla MQWQR -rekord kolejki obciążenia klastra.

Wartości początkowe i deklaracje językowe dla MQWQR

Wartości początkowe oraz deklaracje języka C i High Level Assembler dla MQWQR -rekord kolejki obciążenia klastra.

Tabela 832. Pola w MQWQR

Nazwa pola	Nazwa stałej	Wartość stałej
StrucId	MQWQR_STRUC_ID_ARRAY	'WQR¬'
Version	MQWQR_VERSION_1	1
StrucLength	MQWQR_CURRENT_LENGTH [3]	212
QFlags	Brak	0
QName	Brak	""
QMgrIdentifier	Brak	""
ClusterRecOffset	Brak	0
QType	Brak	0
QDesc	Brak	""
DefBind	Brak	0
DefPersistence	Brak	0
DefPriority	Brak	0
InhibitPut	Brak	0
CLWLQueuePriority	Brak	0
CLWLQueueRank	Brak	0
DefPutResponse	Brak	1

Uwagi:

1. Symbol ¬ reprezentuje pojedynczy znak odstępu.
2. W języku programowania C zmienna makra MQWQR_DEFAULT zawiera wartości domyślne. Użyj go w następujący sposób, aby podać wartości początkowe dla pól w strukturze:

```
MQWQR MyWQR = {MQWQR_DEFAULT};
```

3. Wartości początkowe celowo ustawiły długość struktury na długość bieżącej wersji, a nie na wersję 1 struktury.

Deklaracja C

```
typedef struct tagMQWQR {
  MQCHAR4   StrucId;           /* Structure identifier */
  MQLONG    Version;           /* Structure version number */
  MQLONG    StrucLength;       /* Length of MQWQR structure */
  MQLONG    QFlags;            /* Queue flags */
  MQCHAR48  QName;             /* Queue name */
  MQCHAR48  QMgrIdentifier;    /* Queue manageridentifier */
  MQLONG    ClusterRecOffset;  /* Offset of first cluster record */
  MQLONG    QType;             /* Queue type */
  MQCHAR64  QDesc;             /* Queue description */
  MQLONG    DefBind;           /* Default binding */
  MQLONG    DefPersistence;    /* Default message persistence */
  MQLONG    DefPriority;       /* Default message priority */
  MQLONG    InhibitPut;        /* Whether put operations on the queue
                                  are allowed */
```

```
  /* version 2 */
  MQLONG    CLWLQueuePriority; /* Queue priority */
  MQLONG    CLWLQueueRank;     /* Queue rank */
  /* version 3 */
  MQLONG    DefPutResponse;    /* Default put response */
};
```

High Level Assembler

```
MQWQR                           DSECT
MQWQR_STRUCID                   DS   CL4      Structure identifier
MQWQR_VERSION                   DS   F        Structure version number
MQWQR_STRUCLENGTH               DS   F        Length of MQWQR structure
MQWQR_QFLAGS                    DS   F        Queue flags
MQWQR_QNAME                     DS   CL48     Queue name
MQWQR_QMGRIDENTIFIER            DS   CL48     Queue manager identifier
MQWQR_CLUSTERRECOFFSET          DS   F        Offset of first cluster
*                                            record
MQWQR_QTYPE                     DS   F        Queue type
MQWQR_QDESC                     DS   CL64     Queue description
MQWQR_DEFBIND                   DS   F        Default binding
MQWQR_DEFPERSISTENCE            DS   F        Default message persistence
MQWQR_DEFPRIORITY               DS   F        Default message priority
MQWQR_INHIBITPUT                DS   F        Whether put operations on
*                                            the queue are allowed
MQWQR_DEFPUTRESPONSE            DS   F        Default put response
MQWQR_LENGTH                    EQU  *-MQWQR  Length of structure
                                ORG  MQWQR
MQWQR_AREA                      DS   CL(MQWQR_LENGTH)
```

Odsyłacze pokrewne

Pola w produkcie MQWQR -Struktura rekordu kolejki obciążenia klastra
Opis pól pakietu MQWQR -Struktura rekordu kolejki obciążenia klastra.

MQWCR -Struktura rekordu klastra obciążenia klastra

Poniższa tabela zawiera podsumowanie pól w strukturze rekordu obciążenia klastra MQWCR .

Tabela 833. Pola w MQWCR		
Pole	**Opis**	**Strona**
ClusterName	Nazwa klastra	ClusterName
ClusterRecOffset	Przesunięcie następnego rekordu klastra (MQWCR)	ClusterRecprzesunięcie
ClusterFlags	Flagi klastra	ClusterFlags

Struktura rekordu klastra obciążenia klastra zawiera informacje o klastrze. Dla każdego klastra, do którego należy kolejka docelowa, istnieje jedna struktura rekordu klastra obciążenia klastra.

Struktura rekordu klastra obciążenia klastra jest obsługiwana we wszystkich środowiskach.

Odsyłacze pokrewne

MQ_CLUSTER_WORKLOAD_EXIT -opis połączenia
Wyjście obciążenia klastra jest wywoływane przez menedżer kolejek w celu skierowania komunikatu do dostępnego menedżera kolejek.

MQXCLWLN -nawigowanie po rekordach obciążenia klastra
Wywołanie MQXCLWLN jest używane do przechodzenia między łańcuchami rekordów MQWDR, MQWQRi MQWCR zapisanymi w pamięci podręcznej klastra.

MQWXP -Struktura parametru wyjścia obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze parametru wyjścia obciążenia klastra MQWXP .

MQWDR-Struktura rekordu docelowego obciążenia klastra
Poniższa tabela zawiera podsumowanie pól w strukturze rekordu docelowego obciążenia klastra MQWDR .

MQWQR -Struktura rekordu kolejki obciążenia klastra

Poniższa tabela zawiera podsumowanie pól w strukturze rekordu kolejki obciążenia klastra MQWQR .

Pola w strukturze rekordu klastra obciążenia klastra MQWCR .

Opis pól pakietu MQWCR -Struktura rekordu klastra obciążenia klastra.

ClusterName (MQCHAR48) -wejście
Nazwa klastra, do którego należy instancja kolejki docelowej będącej właścicielem struktury MQWCR . Instancja kolejki docelowej jest opisana przez strukturę MQWDR .

- Długość łańcucha ClusterName wynosi MQ_CLUSTER_NAME_LENGTH.

ClusterRecPrzesunięcie (MQLONG) -wejście
Przesunięcie logiczne następnej struktury pakietu MQWCR .

- Jeśli nie ma więcej struktur MQWCR , ClusterRecprzesunięcie wynosi zero.
- Przesunięcie jest mierzone w bajtach od początku struktury MQWCR .

ClusterFlags (MQLONG) -wejście
Flagi klastra wskazują właściwości menedżera kolejek identyfikowanego przez strukturę MQWCR . Zdefiniowane są następujące opcje:

MQQMF_REPOSITORY_Q_MGR
Miejsce docelowe jest menedżerem kolejek repozytorium pełnego.

MQQMF_CLUSSDR_USER_DEFINED
Kanał nadawczy klastra został zdefiniowany ręcznie.

MQQMF_CLUSSDR_AUTO_DEFINED
Kanał nadawczy klastra został zdefiniowany automatycznie.

MQQMF_AVAILABLE
Docelowy menedżer kolejek jest dostępny na potrzeby odbierania komunikatów.

Inne wartości
Inne flagi w tym polu mogą być ustawiane przez menedżer kolejek na potrzeby wewnętrzne.

Odsyłacze pokrewne

Wartości początkowe i deklaracje językowe pakietu MQWCR
Wartości początkowe oraz deklaracje języka C i High Level Assembler dla MQWCR -Struktura rekordu klastra obciążenia klastra.

Wartości początkowe i deklaracje językowe pakietu MQWCR

Wartości początkowe oraz deklaracje języka C i High Level Assembler dla MQWCR -Struktura rekordu klastra obciążenia klastra.

Tabela 834. Pola w MQWCR

Nazwa pola	Nazwa stałej	Wartość stałej
ClusterName	Brak	" "
ClusterRecOffset	Brak	0
ClusterFlags	Brak	0

Deklaracja C

```
typedef struct tagMQWCR {
  MQCHAR48  ClusterName;       /* Cluster name */
  MQLONG    ClusterRecOffset;  /* Offset of next cluster record */
  MQLONG    ClusterFlags;      /* Cluster flags */
};
```

High Level Assembler

```
MQWCR                           DSECT
MQWCR_CLUSTERNAME               DS   CL48     Cluster name
MQWCR_CLUSTERRECOFFSET          DS   F        Offset of next cluster
*                                            record
MQWCR_CLUSTERFLAGS              DS   F        Cluster flags
MQWCR_LENGTH                    EQU  *-MQWCR  Length of structure
                                ORG  MQWCR
MQWCR_AREA                      DS   CL(MQWCR_LENGTH)
```

Odsyłacze pokrewne

Pola w strukturze rekordu klastra obciążenia klastra MQWCR .
Opis pól pakietu MQWCR -Struktura rekordu klastra obciążenia klastra.

Odwołanie do wyjścia funkcji API

Ta sekcja zawiera informacje uzupełniające dotyczące głównie procedur zewnętrznych interfejsu API napisanych przez programistę.

Ogólne uwagi dotyczące użycia

uwagi:

1. Wszystkie funkcje wyjścia mogą wywoływać wywołanie MQXEP. To wywołanie zostało zaprojektowane specjalnie do użytku z funkcji wyjścia interfejsu API.
2. Funkcja MQ_INIT_EXIT nie może wywołać żadnych wywołań produktu MQ innych niż MQXEP.
3. Nie można wywołać wywołania MQDISC dla bieżącego połączenia.
4. Jeśli funkcja wyjścia wywoła wywołanie MQCONN lub wywołanie MQCONNX z opcją MQCNO_HANDLE_SHARE_NONE, wywołanie zostanie zakończone z kodem przyczyny MQRC_ALREADY_CONNECTED, a zwrócony uchwyt będzie taki sam, jak uchwyt przekazany do wyjścia jako parametr.
5. Generalnie, gdy funkcja wyjścia funkcji API wysyła wywołanie MQI, wyjścia funkcji API nie są wywoływane rekurencyjnie. Jeśli jednak funkcja wyjścia wysyła wywołanie MQCONNX z opcjami MQCNO_HANDLE_SHARE_BLOCK lub MQCNO_HANDLE_SHARE_NO_BLOCK, wywołanie zwraca nowy uchwyt współużytkowany. Dzięki temu pakiet obsługi wyjścia ma własny uchwyt połączenia, a tym samym jednostkę pracy, która jest niezależna od jednostki pracy aplikacji. Pakiet obsługi wyjścia może używać tego uchwytu do umieszczania i pobierania komunikatów w obrębie własnej jednostki pracy oraz do zatwierdzania lub wycofywania tej jednostki pracy. Wszystko to można wykonać bez wpływu na jednostkę pracy aplikacji w jakikolwiek sposób.

   Ponieważ funkcja wyjścia używa uchwytu połączenia, który różni się od uchwytu używanego przez aplikację, wywołania produktu MQ wywołane przez funkcję wyjścia powodują wywołanie odpowiednich funkcji wyjścia funkcji API. Dlatego funkcje wyjścia mogą być wywoływane rekurencyjnie. Należy zauważyć, że zarówno pole *ExitUserArea* w MQAXP, jak i obszar łańcucha wyjścia mają zasięg uchwytu połączenia. W związku z tym funkcja wyjścia nie może użyć tych obszarów do zasygnalizowania innej instancji samej siebie wywołanej rekurencyjnie, że jest ona już aktywna.
6. Funkcje wyjścia mogą również wstawiać i otrzymywać komunikaty w obrębie jednostki pracy aplikacji. Gdy aplikacja zatwierdza lub wycofuje jednostkę pracy, wszystkie komunikaty w jednostce pracy są zatwierdzane lub wycofywane razem, niezależnie od tego, kto umieścił je w jednostce pracy (funkcja aplikacji lub wyjścia). Jednak wyjście może spowodować, że aplikacja przekroczy limity systemowe wcześniej niż w innym przypadku (na przykład przez przekroczenie maksymalnej liczby niezatwierdzonych komunikatów w jednostce pracy).

   Jeśli funkcja wyjścia używa jednostki pracy aplikacji w ten sposób, zwykle powinna unikać wywołania MQCMIT, ponieważ powoduje to zatwierdzenie jednostki pracy aplikacji i może mieć negatywny wpływ na poprawne działanie aplikacji. Jeśli jednak funkcja wyjścia napotka poważny błąd, który uniemożliwia zatwierdzenie jednostki pracy (na przykład błąd powodujący umieszczenie komunikatu

jako części jednostki pracy aplikacji), może być konieczne wywołanie wywołania MQBACK. Po wywołaniu komendy MQBACK należy upewnić się, że granice jednostki pracy aplikacji nie zostały zmienione. W takiej sytuacji funkcja wyjścia musi ustawić odpowiednie wartości, aby kod zakończenia MQCC_WARNING i kod przyczyny MQRC_BACKED_OUT były zwracane do aplikacji, dzięki czemu aplikacja może wykryć, że jednostka pracy została wycofana.

Jeśli funkcja wyjścia używa uchwytu połączenia aplikacji do wywoływania wywołań produktu MQ , same te wywołania nie powodują dalszych wywołań funkcji wyjścia interfejsu API.

7. Jeśli funkcja wyjścia MQXR_BEFORE zakończy się nieprawidłowo, może być możliwe odtworzenie menedżera kolejek po awarii. Jeśli jest to możliwe, menedżer kolejek kontynuuje przetwarzanie tak, jakby funkcja wyjścia zwróciła błąd MQXCC_FAILED. Jeśli nie można odzyskać menedżera kolejek, aplikacja zostanie zakończona.
8. Jeśli funkcja wyjścia MQXR_AFTER zakończy działanie nieprawidłowo, menedżer kolejek może być w stanie odzyskać sprawność po awarii. Jeśli jest to możliwe, menedżer kolejek kontynuuje przetwarzanie tak, jakby funkcja wyjścia zwróciła błąd MQXCC_FAILED. Jeśli nie można odzyskać menedżera kolejek, aplikacja zostanie zakończona. Należy pamiętać, że w tym drugim przypadku komunikaty pobierane poza jednostką pracy są tracone (jest to taka sama sytuacja, jak w przypadku awarii aplikacji natychmiast po usunięciu komunikatu z kolejki).
9. Proces MCA wykonuje zatwierdzanie dwufazowe.

   Jeśli wyjście funkcji API przechwyci komunikat MQCMIT z przygotowanego procesu MCA i podejmie próbę wykonania działania w jednostce pracy, działanie nie powiedzie się i zostanie zwrócony kod przyczyny MQRC_UOW_NOT_AVAILABLE.
10. Jeśli dostępnych jest wiele instalacji produktu IBM MQ , należy użyć wyjść napisanych dla wcześniejszej wersji produktu IBM MQ, ponieważ nowe funkcje dodane w nowszej wersji mogą nie działać z wcześniejszymi wersjami. Więcej informacji na temat zmian między wersjami zawiera sekcja Co zmieniono w produkcie IBM MQ 8.0.

Struktura parametrów wyjścia funkcji API języka IBM MQ (MQAXP)

Struktura MQAXP, zewnętrzny blok kontrolny, jest używana jako parametr wejściowy lub wyjściowy wyjścia funkcji API. Ten temat zawiera również informacje na temat przetwarzania funkcji wyjścia przez menedżery kolejek.

MQAXP ma następującą deklarację 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 */
};
```

Podczas wywoływania funkcji w wyjściu funkcji API przekazywana jest następująca lista parametrów:

StrucId (MQCHAR4)-dane wejściowe
Identyfikator struktury parametru wyjścia o wartości:

```
MQAXP_STRUC_ID.
```

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

Wersja (MQLONG)-wejście
Numer wersji struktury o wartości:

MQAXP_VERSION_1
Struktura parametru wyjścia funkcji API w wersji 1.

MQAXP_VERSION_2
Struktura parametru wyjścia funkcji API w wersji 2.

MQAXP_CURRENT_VERSION
Bieżący numer wersji struktury parametru wyjścia funkcji API.

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

ExitId (MQLONG)-wejście
Identyfikator wyjścia, ustawiony na wejściu do procedury wyjścia, wskazujący typ wyjścia:

MQXT_API_EXIT
Wyjście funkcji API.

ExitReason (MQLONG)-wejście
Przyczyna wywołania wyjścia, ustawiona na wejściu dla każdej funkcji wyjścia:

MQXR_CONNECTION (połączenie MQXR)
Wyjście jest wywoływane w celu zainicjowania przed wywołaniem MQCONN lub MQCONNX lub zakończenia po wywołaniu MQDISC.

MQXR_BEFORE,
Wyjście jest wywoływane przed wykonaniem wywołania funkcji API lub przed przekształceniem danych w MQGET.

MQXR_PO
Wyjście jest wywoływane po wykonaniu wywołania funkcji API.

ExitResponse (MQLONG)-dane wyjściowe
Odpowiedź z wyjścia, inicjowana przy wejściu do każdej funkcji wyjścia w celu:

MQXCC_OK
Kontynuuj normalnie.

To pole musi być ustawione przez funkcję wyjścia, aby poinformować menedżera kolejek o wyniku wykonania funkcji wyjścia. Wartość musi być jedną z następujących wartości:

MQXCC_OK
Funkcja wyjścia zakończyła się pomyślnie. Kontynuuj normalnie.

Ta wartość może zostać ustawiona przez wszystkie funkcje wyjścia MQXR_ *. Komenda ExitResponse2 jest używana do określenia, czy funkcje wyjścia mają być wywoływane później w łańcuchu.

MQXCC-NIEPOWODZENIE
Funkcja wyjścia nie powiodła się z powodu błędu.

Ta wartość może zostać ustawiona przez wszystkie funkcje wyjścia MQXR_ *. Menedżer kolejek ustawia wartość CompCode na MQCC_FAILED, a przyczyną jest:

- MQRC_API_EXIT_INIT_ERROR, jeśli funkcją jest MQ_INIT_EXIT
- MQRC_API_EXIT_TERM_ERROR, jeśli funkcją jest MQ_TERM_EXIT
- MQRC_API_EXIT_ERROR dla wszystkich innych funkcji wyjścia

Zestaw wartości może zostać zmieniony przez funkcję wyjścia w dalszej części łańcucha.

Odpowiedź ExitResponse2 jest ignorowana; menedżer kolejek kontynuuje przetwarzanie tak, jakby została zwrócona wartość MQXR2_SUPPRESS_CHAIN .

MQXCC_SUPPRESS_FUNCTION,
Pomijanie funkcji API IBM MQ .

Tę wartość można ustawić tylko za pomocą funkcji wyjścia MQXR_BEFORE. Pomija wywołanie funkcji API. Jeśli jest ona zwracana przez komendę MQ_DATA_CONV_ON_GET_EXIT, konwersja danych jest pomijana. Menedżer kolejek ustawia wartość CompCode na MQCC_FAILED i Reason na MQRC_SUPPRESSED_BY_EXIT, ale wartości można zmienić za pomocą funkcji obsługi wyjścia w dalszej części łańcucha. Pozostałe parametry dla wywołania pozostają takie, jak wyjście je opuściło. Komenda ExitResponse2 jest używana do określenia, czy funkcje wyjścia mają być wywoływane później w łańcuchu.

Jeśli ta wartość jest ustawiana przez funkcję wyjścia MQXR_AFTER lub MQXR_CONNECTION, menedżer kolejek kontynuuje przetwarzanie tak, jakby została zwrócona wartość MQXCC_FAILED.

MQXCC_SKIP_FUNCTION,
Pomiń funkcję API IBM MQ .

Tę wartość można ustawić tylko za pomocą funkcji wyjścia MQXR_BEFORE. Pomija wywołanie funkcji API. Jeśli jest ona zwracana przez komendę MQ_DATA_CONV_ON_GET_EXIT, konwersja danych jest pomijana. Funkcja wyjścia musi ustawić CompCode i Reason na wartości, które mają zostać zwrócone do aplikacji, ale zestaw wartości może zostać zmieniony przez funkcję wyjścia później w łańcuchu. Pozostałe parametry dla wywołania pozostają takie, jak wyjście je opuściło. Komenda ExitResponse2 jest używana do określenia, czy funkcje wyjścia mają być wywoływane później w łańcuchu.

Jeśli ta wartość jest ustawiana przez funkcję wyjścia MQXR_AFTER lub MQXR_CONNECTION, menedżer kolejek kontynuuje przetwarzanie tak, jakby została zwrócona wartość MQXCC_FAILED.

MQXCC_SUPPRESS_EXIT
Pomijaj wszystkie funkcje wyjścia należące do zestawu wyjść.

Tę wartość można ustawić tylko za pomocą funkcji wyjścia MQXR_BEFORE i MQXR_AFTER. Pomija *wszystkie* kolejne wywołania funkcji obsługi wyjścia należące do tego zestawu wyjść dla tego połączenia logicznego. To obejście jest kontynuowane do momentu wystąpienia żądania logicznego rozłączenia, gdy funkcja MQ_TERM_EXIT jest wywoływana z parametrem ExitReason o wartości MQXR_CONNECTION.

Funkcja wyjścia musi ustawić CompCode i Reason na wartości, które mają zostać zwrócone do aplikacji, ale zestaw wartości może zostać zmieniony przez funkcję wyjścia później w łańcuchu. Pozostałe parametry dla wywołania pozostają takie, jak wyjście je opuściło. Odpowiedź ExitResponse2 jest ignorowana.

Jeśli ta wartość jest ustawiona przez funkcję wyjścia MQXR_CONNECTION, menedżer kolejek kontynuuje przetwarzanie tak, jakby została zwrócona wartość MQXCC_FAILED.

Informacje na temat interakcji między elementami ExitResponse i ExitResponse2oraz jej wpływu na przetwarzanie wyjścia zawiera sekcja "W jaki sposób menedżery kolejek przetwarzają funkcje wyjścia" na stronie 1616.

ExitResponse2 (MQLONG)-wyjście
Jest to dodatkowy kod odpowiedzi wyjścia, który kwalifikuje podstawowy kod odpowiedzi wyjścia dla funkcji wyjścia MQXR_BEFORE. Jest on inicjowany w następujący sposób:

```
MQXR2_DEFAULT_CONTINUATION
```

na wejściu do funkcji wyjścia wywołania funkcji API IBM MQ . Można ją następnie ustawić na jedną z następujących wartości:

MQXR2_DEFAULT_CONTINUATION
Określa, czy kontynuować od następnego wyjścia w łańcuchu, w zależności od wartości ExitResponse.

Jeśli ExitResponse to MQXCC_SUPPRESS_FUNCTION lub MQXCC_SKIP_FUNCTION, należy pominąć funkcje wyjścia później w łańcuchu MQXR_BEFORE i zgodne funkcje wyjścia w łańcuchu

MQXR_AFTER. Wywołaj funkcje wyjścia w łańcuchu MQXR_AFTER, które odpowiadają funkcjom wyjścia wcześniej w łańcuchu MQXR_BEFORE.

W przeciwnym razie wywołaj następne wyjście w łańcuchu.

MQXR2_SUPPRESS_CHAIN
Zablokuj łańcuch.

Pomiń funkcje wyjścia później w łańcuchu MQXR_BEFORE i zgodne funkcje wyjścia w łańcuchu MQXR_AFTER dla tego wywołania funkcji API. Wywołaj funkcje wyjścia w łańcuchu MQXR_AFTER, które odpowiadają funkcjom wyjścia wcześniej w łańcuchu MQXR_BEFORE.

MQXR2_CONTINUE_CHAIN
Przejdź do następnego wyjścia w łańcuchu.

Informacje na temat interakcji między elementami ExitResponse i ExitResponse2oraz jej wpływu na przetwarzanie wyjścia zawiera sekcja "W jaki sposób menedżery kolejek przetwarzają funkcje wyjścia" na stronie 1616.

Feedback (MQLONG)-wejście/wyjście
Przekazywanie kodów sprzężenia zwrotnego między wywołaniami funkcji wyjścia. Jest on inicjowany w następujący sposób:

```
MQFB_NONE (0)
```

przed wywołaniem pierwszej funkcji pierwszego wyjścia w łańcuchu.

Wyjścia mogą ustawić w tym polu dowolną wartość, w tym dowolną poprawną wartość MQFB_ * lub MQRC_ *. Wyjścia mogą również ustawić w tym polu zdefiniowaną przez użytkownika wartość sprzężenia zwrotnego z zakresu od MQFB_APPL_FIRST do MQFB_APPL_LAST.

APICallerType (MQLONG)-wejście
Typ programu wywołującego API, który wskazuje, czy program wywołujący API IBM MQ jest zewnętrzny, czy wewnętrzny względem menedżera kolejek: MQXACT_EXTERNAL lub MQXACT_INTERNAL.

Obszar ExitUser(MQBYTE16)-wejście/wyjście
Obszar użytkownika dostępny dla wszystkich wyjść powiązanych z konkretnym obiektem ExitInfo. Przed wywołaniem pierwszej funkcji wyjścia (MQ_INIT_EXIT) dla hconn jest inicjowana wartość MQXUA_NONE (zera binarne dla obszaru ExitUser). Od tego momentu wszystkie zmiany wprowadzone w tym polu przez funkcję wyjścia są zachowywane w wywołaniach funkcji tego samego wyjścia.

To pole jest wyrównane do wielokrotności 4 MQLONGs.

Wyjścia mogą również zakotwiczyć dowolną pamięć masową, którą przydzielają z tego obszaru.

Dla każdego hconn każde wyjście w łańcuchu wyjść ma inny obszar ExitUser. Obszar ExitUsernie może być współużytkowany przez wyjścia w łańcuchu, a zawartość obszaru ExitUserdla jednego wyjścia nie jest dostępna dla innego wyjścia w łańcuchu.

W przypadku programów w języku C stała MQXUA_NONE_ARRAY jest również definiowana z taką samą wartością, jak MQXUA_NONE, ale jako tablica znaków zamiast łańcucha.

Długość tego pola jest określona przez wartość MQ_EXIT_USER_AREA_LENGTH.

ExitData (MQCHAR32)-dane wejściowe
Dane wyjścia, ustawione na wejściu dla każdej funkcji wyjścia na 32 znaki danych wyjścia, które są udostępniane w wyjściu. Jeśli w polu wyjścia nie zostanie zdefiniowana żadna wartość, to pole będzie puste.

Długość tego pola jest określona przez wartość MQ_EXIT_DATA_LENGTH.

ExitInfoNazwa (MQCHAR48)-wejście
Nazwa informacji o wyjściu, ustawiana na wejściu dla każdej funkcji wyjścia na nazwę ApiExit_name określoną w definicjach wyjścia w sekcjach.

ExitPDArea (MQBYTE48)-wejście/wyjście
Obszar określania problemu, inicjowany przez MQXPDA_NONE (binarne zera dla długości pola) dla każdego wywołania funkcji wyjścia.

W przypadku programów w języku C stała MQXPDA_NONE_ARRAY jest również zdefiniowana z taką samą wartością, jak MQXPDA_NONE, ale jako tablica znaków zamiast łańcucha.

Procedura obsługi wyjścia zawsze zapisuje ten obszar w danych śledzenia IBM MQ na końcu wyjścia, nawet jeśli funkcja zakończyła się pomyślnie.

Długość tego pola jest określona przez wartość MQ_EXIT_PD_AREA_LENGTH.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek, z którym połączona jest aplikacja, który wywołał wyjście w wyniku przetwarzania wywołania funkcji API języka IBM MQ .

Jeśli nazwa menedżera kolejek podana w wywołaniach MQCONN lub MQCONNX jest pusta, w tym polu nadal jest ustawiona nazwa menedżera kolejek, z którym połączona jest aplikacja, bez względu na to, czy aplikacja jest serwerem, czy klientem.

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

Długość tego pola jest określona przez wartość MQ_Q_MGR_NAME_LENGTH.

ExitChainAreaPtr (PMQACH)-wejście/wyjście
Służy do przesyłania danych między wywołaniami różnych wyjść w łańcuchu. Jest on ustawiany na wskaźnik NULL przed wywołaniem pierwszej funkcji (MQ_INIT_EXIT z parametrem ExitReason MQXR_CONNECTION) pierwszego wyjścia w łańcuchu wyjść. Wartość zwracana przez wyjście w jednym wywołaniu jest przekazywana do następnego wywołania.

Więcej informacji na temat korzystania z obszaru łańcucha wyjściowego zawiera sekcja “Obszar łańcucha wyjścia i nagłówek obszaru łańcucha wyjścia (MQACH)” na stronie 1619 .

Hconfig (MQHCONFIG)-wejście
Uchwyt konfiguracji reprezentujący inicjowany zestaw funkcji. Ta wartość jest generowana przez menedżer kolejek w funkcji MQ_INIT_EXIT, a następnie jest przekazywana do funkcji wyjścia funkcji API. Jest on ustawiany na wejściu do każdej funkcji wyjścia.

Można użyć Hconfig jako wskaźnika do struktury MQIEP w celu wykonania wywołań MQI i DCI. Przed użyciem parametru HConfig jako wskaźnika do struktury MQIEP należy sprawdzić, czy pierwsze 4 bajty konfiguracji HConfig są zgodne z wartością StrucId struktury MQIEP.

Funkcja (MQLONG)-wejście
Identyfikator funkcji, poprawne wartości to stałe MQXF_ * opisane w sekcji “Stałe zewnętrzne” na stronie 1621.

Procedura obsługi wyjścia ustawia w tym polu poprawną wartość na wejściu do każdej funkcji wyjścia, w zależności od wywołania funkcji API IBM MQ , które spowodowało wywołanie wyjścia.

ExitMsgUchwyt (MQHMSG)-wejście/wyjście
Jeśli funkcja to MQXF_GET, a parametr ExitReason ma wartość MQXR_AFTER, w tym polu jest zwracany poprawny uchwyt komunikatu, który umożliwia wyjściu funkcji API dostęp do pól deskryptora komunikatu i innych właściwości zgodnych z łańcuchem ExitProperties określonym w strukturze MQXEPO podczas rejestrowania wyjścia funkcji API.

Wszystkie właściwości deskryptora inne niż komunikat, które są zwracane w uchwycie ExitMsg, nie będą dostępne w polu MsgHandle w strukturze MQGMO, jeśli został określony, ani w danych komunikatu.

Jeśli funkcja ma wartość MQXF_GET, a parametr ExitReason ma wartość MQXR_BEFORE, to jeśli program obsługi wyjścia ustawi w tym polu wartość MQHM_NONE, nie będzie zapełniał właściwości uchwytu ExitMsg.

To pole nie jest ustawiane, jeśli wersja jest wcześniejsza niż MQAXP_VERSION_2.

W jaki sposób menedżery kolejek przetwarzają funkcje wyjścia

Przetwarzanie wykonywane przez menedżer kolejek w przypadku powrotu z funkcji wyjścia zależy zarówno od wartości ExitResponse , jak i ExitResponse2.

Tabela 835 na stronie 1616 zawiera podsumowanie możliwych kombinacji i ich efektów dla funkcji wyjścia MQXR_BEFORE:

- Kto ustawia parametry CompCode i Reason wywołania interfejsu API
- Określa, czy są wywoływane pozostałe funkcje wyjścia w łańcuchu MQXR_BEFORE i zgodne funkcje wyjścia w łańcuchu MQXR_AFTER.
- Informacja o tym, czy wywołanie interfejsu API jest wywoływane

Dla funkcji wyjścia MQXR_AFTER:

- CompCode i Reason są ustawione w taki sam sposób, jak MQXR_BEFORE
- Odpowiedź ExitResponse2 jest ignorowana (pozostałe funkcje wyjścia w łańcuchu MQXR_AFTER są zawsze wywoływane)
- MQXCC_SUPPRESS_FUNCTION i MQXCC_SKIP_FUNCTION są niepoprawne

Dla funkcji wyjścia MQXR_CONNECTION:

- CompCode i Reason są ustawione w taki sam sposób, jak MQXR_BEFORE
- ExitResponse2 jest ignorowane
- MQXCC_SUPPRESS_FUNCTION, MQXCC_SKIP_FUNCTION, MQXCC_SUPPRESS_EXIT są niepoprawne

We wszystkich przypadkach, gdy wyjście lub menedżer kolejek ustawia wartość CompCode i Reason, zestaw wartości może zostać zmieniony przez wyjście wywołane później lub przez wywołanie funkcji API (jeśli wywołanie funkcji API zostanie wywołane później).

Tabela 835. Przetwarzanie wyjścia MQXR_BEFORE

Wartość ExitResponse	**CompCode i przyczyna ustawiona przez**	**Wartość łańcucha ExitResponse2 (domyślna kontynuacja)**	**Wartość ExitResponse2 (domyślna kontynuacja) API**
MQXCC_OK	exit	Y	Y
MQXCC_SUPPRESS_EXIT	exit	Y	Y
MQXCC_SUPPRESS_FUNCTION,	menedżer kolejek	N	N
MQXCC_SKIP, FUNKCJA	exit	N	N
MQXCC-NIEPOWODZENIE	menedżer kolejek	N	N

Jak klienty przetwarzają funkcje obsługi wyjścia

Ogólnie rzecz biorąc, klienty przetwarzają funkcje wyjścia w taki sam sposób, jak aplikacje serwera, a atrybut *QMgrName* w tej strukturze ma zastosowanie bez względu na to, czy funkcja znajduje się na serwerze, czy na kliencie.

Jednak klient nie ma pojęcia o pliku *mqs.ini* , dlatego sekcje *ApiExitCommon* i *APIExitTemplate* nie mają zastosowania. Zastosowanie ma tylko sekcja *ApiExitLocal* , która jest skonfigurowana w pliku *mqclient.ini* .

Struktura kontekstu wyjścia funkcji API języka IBM MQ (MQAXC)

Struktura MQAXC, zewnętrzny blok sterujący, jest używana jako parametr wejściowy wyjścia funkcji API.

MQAXC ma następującą deklarację 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 */
};
```

Parametry MQAXC są następujące:

StrucId (MQCHAR4)-dane wejściowe
Identyfikator struktury kontekstu wyjścia o wartości MQAXC_STRUC_ID. W przypadku programów w języku C zdefiniowana jest również stała MQAXC_STRUC_ID_ARRAY z taką samą wartością jak MQAXC_STRUC_ID, ale jako tablica znaków zamiast łańcucha.

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

Wersja (MQLONG)-wejście
Numer wersji struktury o wartości:

MQAXC_VERSION_2
Numer wersji struktury kontekstu wyjścia.

MQAXC_CURRENT_VERSION
Bieżący numer wersji struktury kontekstu wyjścia.

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

Środowisko (MQLONG)-wejście
Środowisko, z którego wywołano funkcję API IBM MQ , co spowodowało, że funkcja wyjścia była sterowana. Poprawne wartości dla tego pola to:

MQXE_INNY
Ta wartość jest spójna z wywołaniami wyjścia funkcji API, które widzą, czy wyjście jest wywoływane z aplikacji serwera. Oznacza to, że wyjście funkcji API działa bez zmian na kliencie i nie widzi niczego innego.

Jeśli wyjście rzeczywiście wymaga określenia, czy jest uruchomione na kliencie, można to zrobić, sprawdzając pola *ChannelName* i *ChannelDefinition* .

Agent MQXE_MCA
Agent kanału komunikatów

MQXE_MCA_SVRCONN
Agent kanału komunikatów działający w imieniu klienta

SERWER MQXE_COMMAND_SERVER
Serwer komend

MQXE_MQSC
Interpreter komendy runmqsc

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

UserId (MQCHAR12)-dane wejściowe
Identyfikator użytkownika powiązany z aplikacją. W szczególności, w przypadku połączeń klienckich, pole to zawiera identyfikator użytkownika adoptowanego w przeciwieństwie do identyfikatora użytkownika, pod którym działa kod kanału. Jeśli pusty ID użytkownika przepływa z klienta, nie

są wprowadzane żadne zmiany w ID użytkownika, który jest już używany. Oznacza to, że nowy ID użytkownika nie jest adoptowany.

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia. Długość tego pola jest określona przez wartość MQ_USER_ID_LENGTH.

W przypadku klienta jest to identyfikator użytkownika wysyłany z klienta do serwera. Należy zauważyć, że może to nie być efektywny identyfikator użytkownika, dla którego klient jest uruchamiany w menedżerze kolejek, ponieważ może to być konfiguracja MCAUser lub CHLAUTH, która zmienia identyfikator użytkownika.

SecurityId (MQBYTE40)-dane wejściowe
Rozszerzenie identyfikatora użytkownika uruchamiającego aplikację. Jego długość jest określona przez wartość MQ_SECURITY_ID_LENGTH.

W przypadku klienta jest to identyfikator użytkownika wysyłany z klienta do serwera. Należy zauważyć, że może to nie być efektywny identyfikator użytkownika, dla którego klient jest uruchamiany w menedżerze kolejek, ponieważ może to być konfiguracja MCAUser lub CHLAUTH, która zmienia identyfikator użytkownika.

ConnectionName (MQCHAR264)-wejście
Pole nazwy połączenia, ustawione na adres klienta. Na przykład w przypadku protokołu TCP/IP będzie to adres IP klienta.

Długość tego pola jest określona przez wartość MQ_CONN_NAME_LENGTH.

W przypadku klienta jest to adres partnera menedżera kolejek.

LongMCAUserIdLength (MQLONG)-wejście
Długość długiego identyfikatora użytkownika MCA.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawiane na długość identyfikatora użytkownika długiego agenta MCA (lub na wartość zero, jeśli taki identyfikator nie istnieje).

W przypadku klienta jest to długi identyfikator użytkownika klienta.

LongRemoteUserIdLength (MQLONG)-input
Długość długiego identyfikatora użytkownika zdalnego.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawiane na długość długiego identyfikatora użytkownika zdalnego. W przeciwnym razie to pole zostanie ustawione na wartość zero.

W przypadku klienta należy ustawić wartość tego pola na zero.

LongMCAUserIdPtr (MQPTR)-wejście
Adres długiego identyfikatora użytkownika MCA.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawiane na adres identyfikatora użytkownika długiego agenta MCA (lub na wskaźnik pusty, jeśli taki identyfikator nie istnieje).

W przypadku klienta jest to długi identyfikator użytkownika klienta.

LongRemoteUserIdPtr (MQPTR)-input
Adres długiego identyfikatora użytkownika zdalnego.

Gdy agent MCA łączy się z menedżerem kolejek, to pole jest ustawiane na adres długiego zdalnego identyfikatora użytkownika (lub na wskaźnik pusty, jeśli taki identyfikator nie istnieje).

W przypadku klienta należy ustawić wartość tego pola na zero.

ApplName (MQCHAR28)-dane wejściowe
Nazwa aplikacji lub komponentu, który wywołał funkcję API IBM MQ .

Reguły generowania ApplName są takie same, jak w przypadku generowania nazwy domyślnej dla wywołania MQPUT.

Wartość tego pola można znaleźć, sprawdzając nazwę programu w systemie operacyjnym. Jego długość jest określona przez wartość MQ_APPL_NAME_LENGTH.

ApplType (MQLONG)-dane wejściowe
Typ aplikacji lub komponentu, który wywołał funkcję API IBM MQ .

Wartością jest MQAT_DEFAULT dla platformy, na której aplikacja jest kompilowana, lub jest ona równa jednej ze zdefiniowanych wartości MQAT_ *.

Procedura obsługi wyjścia ustawia to pole na wejściu do każdej funkcji wyjścia.

ProcessId (MQPID)-dane wejściowe
Identyfikator procesu systemu operacyjnego.

Jeśli ma to zastosowanie, procedura obsługi wyjścia ustawia to pole przy wejściu do każdej funkcji wyjścia.

ThreadId (MQTID)-wejście
Identyfikator wątku MQ . Jest to ten sam identyfikator, który jest używany w zrzutach MQ śledzenia i FFST , ale może być inny niż identyfikator wątku systemu operacyjnego.

Jeśli ma to zastosowanie, procedura obsługi wyjścia ustawia to pole przy wejściu do każdej funkcji wyjścia.

ChannelName (MQCHAR)-wejście
Nazwa kanału, uzupełniona odstępami, jeśli ma zastosowanie i jest znana.

Jeśli nie ma zastosowania, to pole zawiera znaki o kodzie zero.

Reserved1 (MQBYTE4)-wejście
To pole jest zarezerwowane.

ChanneDefinition (PMQCD)-wejście
Wskaźnik do używanej definicji kanału, jeśli ma zastosowanie i jest znany.

Jeśli nie ma zastosowania, to pole zawiera znaki o kodzie zero.

Należy zauważyć, że wskaźnik jest zakończony tylko wtedy, gdy połączenie jest przetwarzane w imieniu kanału IBM MQ i ta definicja kanału została odczytana.

W szczególności definicja kanału nie jest podawana na serwerze, gdy dla kanału jest wykonywane pierwsze wywołanie MQCONN. Ponadto, jeśli wskaźnik jest wypełniony, struktura (i wszelkie struktury podrzędne) wskazywana przez wskaźnik musi być traktowana jako tylko do odczytu; każda aktualizacja struktury prowadziłaby do nieprzewidywalnych wyników i nie jest obsługiwana.

W przypadku klienta pola inne niż te z wartością określoną dla klienta zawierają wartości odpowiednie dla aplikacji klienckiej.

Obszar łańcucha wyjścia i nagłówek obszaru łańcucha wyjścia (MQACH)

W razie potrzeby funkcja wyjścia może uzyskać pamięć masową dla obszaru łańcucha wyjścia i ustawić opcję ExitChainAreaPtr w MQAXP tak, aby wskazywała tę pamięć masową.

Wyjścia (takie same lub różne funkcje wyjścia) mogą uzyskać wiele obszarów łańcucha wyjścia i połączyć je ze sobą. Obszary łańcucha wyjścia mogą być dodawane lub usuwane z tej listy tylko wtedy, gdy są wywoływane z procedury obsługi wyjścia. Dzięki temu nie występują problemy z serializacją spowodowane przez różne wątki, które jednocześnie dodają lub usuwają obszary z listy.

Obszar łańcucha wyjścia musi rozpoczynać się od struktury nagłówka MQACH, której deklaracja w języku C jest następująca:

```
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 */
 };
```

Pola w nagłówku obszaru łańcucha wyjścia są następujące:

StrucId (MQCHAR4)-dane wejściowe
Identyfikator struktury obszaru łańcucha wyjścia z wartością początkową zdefiniowaną przez MQACH_DEFAULT dla identyfikatora MQACH_STRUC_ID.

W przypadku programów w języku C zdefiniowana jest również stała MQACH_STRUC_ID_ARRAY, która ma taką samą wartość jak MQACH_STRUC_ID, ale tablicę znaków zamiast łańcucha.

Wersja (MQLONG)-wejście
Numer wersji struktury, w następujący sposób:

MQACH_VERSION_1
Numer wersji struktury parametru wyjścia.

MQACH_CURRENT_VERSION
Bieżący numer wersji struktury kontekstu wyjścia.

Wartością początkową tego pola, zdefiniowaną w parametrze MQACH_DEFAULT, jest MQACH_CURRENT_VERSION.

Uwaga: Jeśli zostanie wprowadzona nowa wersja tej struktury, układ istniejącej części nie ulegnie zmianie. Funkcje wyjścia muszą sprawdzić, czy numer wersji jest równy lub większy od najniższej wersji zawierającej pola, z których musi korzystać funkcja wyjścia.

StrucLength (MQLONG)-wejście
Długość struktury MQACH. Wyjścia mogą używać tego pola do określania początku danych wyjścia, ustawiając długość struktury utworzonej przez wyjście.

Początkową wartością tego pola, zdefiniowaną w parametrze MQACH_DEFAULT, jest MQACH_CURRENT_LENGTH.

ChainAreaDługość (MQLONG)-wejście
Długość obszaru łańcucha wyjścia, ustawiona na całkowitą długość bieżącego obszaru łańcucha wyjścia, w tym nagłówka MQACH.

Początkowa wartość tego pola, zdefiniowana przez MQACH_DEFAULT, wynosi zero.

ExitInfoNazwa (MQCHAR48)-wejście
Nazwa informacji o wyjściu.

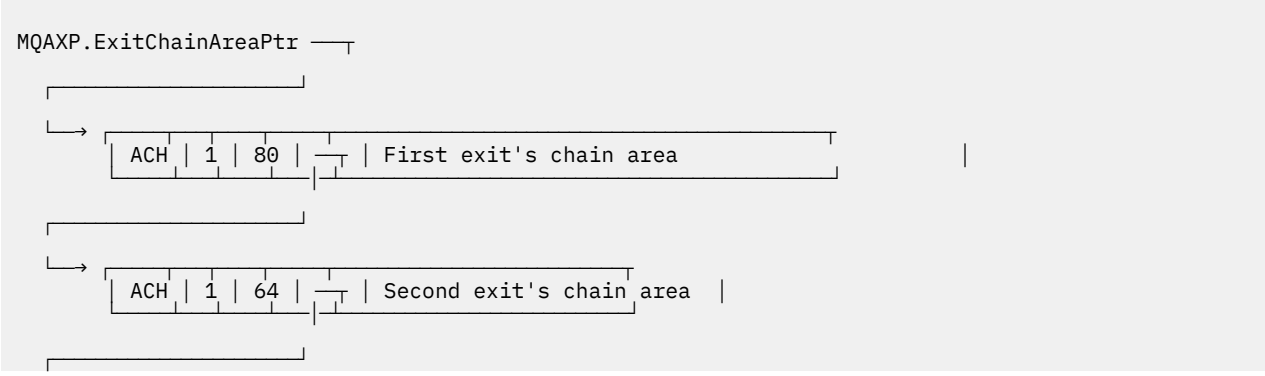
Gdy wyjście tworzy strukturę MQACH, musi zainicjować to pole własną nazwą ExitInfo, aby później ta struktura MQACH mogła zostać znaleziona przez inną instancję tego wyjścia lub przez współpracujące wyjście.

Wartością początkową tego pola, zdefiniowaną w parametrze MQACH_DEFAULT, jest łańcuch o zerowej długości ({""}).

NextChainAreaPtr (PMQACH)-wejście
Wskaźnik do następnego obszaru łańcucha wyjścia z wartością początkową, zdefiniowaną przez MQACH_DEFAULT, wskaźnika pustego (NULL).

Funkcje wyjścia muszą zwolnić pamięć masową dla wszystkich obszarów łańcucha wyjścia, które uzyskują, i manipulować wskaźnikami łańcucha w celu usunięcia ich obszarów łańcucha wyjścia z listy.

Obszar łańcucha wyjściowego może być skonstruowany w następujący sposób:



```
└──→ ... etc.
```

Stałe zewnętrzne

Ten temat zawiera informacje uzupełniające o stałych zewnętrznych dostępnych dla interfejsu API.

Dla wyjść funkcji API dostępne są następujące stałe zewnętrzne:

MQXF_ * (identyfikatory funkcji wyjścia)

```
       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_ * (przyczyny wyjścia)

```
       MQXR_BEFORE                    1       X'00000001'
       MQXR_AFTER                     2       X'00000002'
       MQXR_CONNECTION                3       X'00000003'
```

MQXE_ * (środowiska)

```
       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* _ * (dodatkowe stałe)

```
       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* _ * (stałe o wartości NULL)

```
MQXPDA_NONE                     X'00...00' (48 nulls)
MQXPDA_NONE_ARRAY               '\0','\0',...,'\0','\0'
```

MQXCC_ * (kody zakończenia)

```
MQXCC_FAILED                   -8
```

MQRC_ * (kody przyczyny)

MQRC_API_EXIT_ERROR 2374 X'00000946'
Wywołanie funkcji wyjścia zwróciło niepoprawny kod odpowiedzi lub zakończyło się niepowodzeniem, a menedżer kolejek nie może określić następnego działania do wykonania.

Sprawdź pola ExitResponse i ExitResponse2 w systemie MQAXP, aby określić błędny kod odpowiedzi, i zmień wyjście, aby zwróciło poprawny kod odpowiedzi.

MQRC_API_EXIT_INIT_ERROR 2375 X'00000947'
Menedżer kolejek napotkał błąd podczas inicjowania środowiska wykonawczego dla funkcji wyjścia funkcji API.

MQRC_API_EXIT_TERM_ERROR 2376 X'00000948'
Menedżer kolejek napotkał błąd podczas zamykania środowiska wykonawczego dla funkcji wyjścia funkcji API.

MQRC_EXIT_REASON_ERROR 2377 X'00000949'
Wartość pola ExitReason podana w wywołaniu rejestracji punktu wejścia wyjścia (MQXEP) jest błędna.

Sprawdź wartość w polu ExitReason , aby określić i poprawić błędną wartość przyczyny wyjścia.

MQRC_RESERVED_VALUE_ERROR 2378 X'0000094A'
Wartość w polu Zarezerwowane jest błędna.

Sprawdź wartość w polu Zarezerwowane, aby określić i poprawić wartość Zarezerwowane.

Typy definicji języka C

Ta sekcja zawiera informacje na temat definicji typów powiązanych z wyjściami funkcji API dostępnymi w języku C.

Poniżej przedstawiono typy definicji języka C powiązane z wyjściami funkcji 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;
```

Wywołanie rejestracji punktu wejścia wyjścia (MQXEP)

Te informacje umożliwiają zapoznanie się z prototypem funkcji języka C MQXEP, wywołania języka C MQXEP i MQXEP.

Użyj wywołania MQXEP, aby:

1. Rejestrowanie punktów wywołania przed i po wyjściu funkcji API IBM MQ , w których mają być wywoływane funkcje wyjścia
2. Określ punkty wejścia funkcji wyjścia
3. Wyrejestrowywania punktów wejścia funkcji wyjścia

Zwykle wywołania MQXEP są kodowane w funkcji wyjścia MQ_INIT_EXIT, ale można je określić w każdej kolejnej funkcji wyjścia.

Jeśli do zarejestrowania już zarejestrowanej funkcji wyjścia używane jest wywołanie MQXEP, drugie wywołanie MQXEP zostanie pomyślnie zakończone, zastępując zarejestrowaną funkcję wyjścia.

Jeśli do zarejestrowania funkcji wyjścia o wartości NULL używane jest wywołanie MQXEP, wywołanie MQXEP zostanie pomyślnie zakończone, a funkcja wyjścia zostanie wyrejestrowana.

Jeśli wywołania MQXEP są używane do rejestrowania, wyrejestrowywania i ponownego rejestrowania określonej funkcji wyjścia podczas życia żądania połączenia, poprzednio zarejestrowana funkcja wyjścia jest reaktywowana. Każda pamięć nadal przydzielona i powiązana z tą instancją funkcji wyjścia jest dostępna do użycia przez funkcje wyjścia. (Ta pamięć jest zwykle zwalniana podczas wywoływania funkcji wyjścia zakończenia).

Interfejs do MQXEP to:

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason)
```

gdzie:

Hconfig (MQHCONFIG)-wejście
Uchwyt konfiguracji reprezentujący wyjście funkcji API, które zawiera zestaw inicjowanych funkcji. Ta wartość jest generowana przez menedżer kolejek bezpośrednio przed wywołaniem funkcji MQ_INIT_EXIT i jest przekazywana w MQAXP do każdej funkcji wyjścia funkcji API.

ExitReason (MQLONG)-wejście
Przyczyna, dla której punkt wejścia jest rejestrowany, z następujących powodów:

- Inicjowanie lub kończenie na poziomie połączenia (MQXR_CONNECTION)
- Przed wywołaniem funkcji API IBM MQ (MQXR_BEFORE)
- Po wywołaniu funkcji API języka IBM MQ (MQXR_AFTER)

Funkcja (MQLONG)-wejście
Identyfikator funkcji, poprawne wartości to stałe MQXF_ * (patrz sekcja “Stałe zewnętrzne” na stronie 1621).

EntryPoint (PMQFUNC)-wejście
Adres punktu wejścia dla funkcji wyjścia, która ma zostać zarejestrowana. Wartość NULL wskazuje, że funkcja wyjścia nie została podana lub że poprzednia rejestracja funkcji wyjścia jest wyrejestrowana.

ExitOpts(MQXEPO)
Wyjścia funkcji API mogą określać opcje sterujące rejestrowaniem wyjść funkcji API. Jeśli dla tego pola zostanie określony wskaźnik pusty, zostaną przyjęte wartości domyślne struktury MQXEPO.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny, który kwalifikuje kod zakończenia.

Jeśli kod zakończenia to MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod zakończenia to MQCC_FAILED:

BŁĄD MQRC_HCONFIG_ERROR
(2280, X'8E8') Podany uchwyt konfiguracji jest niepoprawny. Użyj uchwytu konfiguracji z MQAXP.

BŁĄD MQRC_EXIT_REASON_ERROR
(2377, X' 949 ') Podana przyczyna wywołania funkcji wyjścia jest niepoprawna lub nie jest poprawna dla podanego identyfikatora funkcji wyjścia.

Użyj jednej z poprawnych przyczyn wywołania funkcji wyjścia (MQXR_ * wartość) lub użyj poprawnej kombinacji identyfikatora funkcji i przyczyny wyjścia. (Patrz Tabela 836 na stronie 1624.)

BŁĄD FUNKCJI MQRC_FUNCTION_
(2281, X'8E9') Podany identyfikator funkcji nie jest poprawny dla przyczyny wyjścia funkcji API. W poniższej tabeli przedstawiono poprawne kombinacje identyfikatorów funkcji i ExitReasons.

Tabela 836. Poprawne kombinacje identyfikatorów funkcji i ExitReasons

Funkcja	**ExitReason**
MQXF_INIT MQXF_TERM	MQXR_CONNECTION (połączenie MQXR)
MQXF_KONN MQXF_CONNX MQXF_DYSK MQXF_OPEN MQXF_ZAMK. MQXF_PUT1 MQXF_PUT MQXF_GET MQXF_INQ MQXF_SET MQXF_BEGIN MQXF_CMIT MQXF_BACK MQXF_STAT MQXF_CB MQXF_CTL MQXF_CALLBACK MQXF_SUB MQXF_SUBRQ	MQXR_BEFORE, MQXR_PO
MQXF_DATA_CONV_ON_GET	MQXR_BEFORE,

MQRC_RESOURCE_PROBLEM
(2102, X'836 ') Próba zarejestrowania lub wyrejestrowania funkcji obsługi wyjścia nie powiodła się z powodu problemu z zasobem.

BŁĄD MQRC_UNEXPECTED_ERROR
(2195, X'893 ') Próba zarejestrowania lub wyrejestrowania funkcji obsługi wyjścia nie powiodła się nieoczekiwanie.

BŁĄD MQRC_PROPERTY_NAME_ERROR
(2442, X'098A') Niepoprawna nazwa ExitProperties .

BŁĄD MQRC_XEPO_ERROR
(2507, X'09CB') Niepoprawna struktura opcji wyjścia.

Wywołanie języka C MQXEP

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts, &CompCode, &Reason);
```

Deklaracja listy parametrów:

```
    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 */
```

Prototyp funkcji 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 */
```

Funkcje wyjścia

Ta sekcja zawiera ogólne informacje pomocne podczas korzystania z wywołań funkcji i opisuje sposób wywoływania poszczególnych funkcji wyjścia.

Te informacje umożliwiają zrozumienie ogólnych reguł procedur obsługi wyjścia interfejsu API oraz konfigurowanie i czyszczenie środowiska wykonawczego programów zewnętrznych.

Ogólne reguły dla procedur zewnętrznych interfejsu API

Podczas wywoływania procedur wyjścia funkcji API obowiązują następujące reguły ogólne:

- We wszystkich przypadkach funkcje wyjścia interfejsu API są sterowane przed sprawdzaniem poprawności parametrów wywołania interfejsu API i przed sprawdzaniem zabezpieczeń (w przypadku MQCONN, MQCONNX lub MQOPEN).
- Wartości pól wprowadzanych do procedury zewnętrznej i wyprowadzanych z niej są następujące:
  - Na wejściu do *wcześniejszej* funkcji wyjścia funkcji API IBM MQ wartość pola może zostać ustawiona przez aplikację lub przez poprzednie wywołanie funkcji wyjścia.
  - W przypadku danych wyjściowych funkcji wyjścia funkcji API *before* IBM MQ wartość pola może pozostać niezmieniona lub może zostać ustawiona na inną wartość przez funkcję wyjścia.

  - Na wejściu do funkcji wyjścia funkcji API *after* IBM MQ wartość pola może być wartością ustawioną przez menedżer kolejek po przetworzeniu wywołania funkcji API IBM MQ lub może być ustawiona na wartość przez poprzednie wywołanie funkcji wyjścia w łańcuchu funkcji wyjścia.
  - W przypadku danych wyjściowych funkcji wyjścia wywołania funkcji API *after* IBM MQ wartość pola może pozostać niezmieniona lub może zostać ustawiona na inną wartość przez funkcję wyjścia.
- Funkcje wyjścia muszą komunikować się z menedżerem kolejek przy użyciu pól ExitResponse i ExitResponse2 .
- Pola CompCode i kod przyczyny komunikują się z aplikacją. Funkcje menedżera kolejek i wyjścia mogą ustawiać pola CompCode i Kod przyczyny.
- Wywołanie MQXEP zwraca nowe kody przyczyny do funkcji wyjścia, które wywołują MQXEP. Jednak funkcje wyjścia mogą tłumaczyć te nowe kody przyczyn na istniejące kody przyczyn, które mogą być zrozumiałe dla istniejących i nowych aplikacji.
- Każdy prototyp funkcji wyjścia ma podobne parametry do funkcji API z dodatkowym poziomem adresowania pośredniego, z wyjątkiem kodu CompCode i przyczyny.
- Wyjścia interfejsu API mogą wywoływać wywołania interfejsu MQI (z wyjątkiem wywołania MQDISC), ale same te wywołania interfejsu MQI nie wywołują wyjść interfejsu API.

Należy zauważyć, że niezależnie od tego, czy aplikacja znajduje się na serwerze, czy na kliencie, nie można przewidzieć kolejności wywołań wyjścia funkcji API. Wywołanie BEF0RE wyjścia funkcji API może nie następować natychmiast po wywołaniu funkcji AFTER .

Po wywołaniu funkcji BEF0RE może nastąpić kolejne wywołanie funkcji BEF0RE . Na przykład:

PRZED MQCTL
Procedura zwrotna BEFORE
PRZED MQPUT
PO MQPUT
Po wywołaniu zwrotnym
PO OPERACJI MQCTL

lub wersji

PRZED XAOPEN
PRZED MQCONNX
PO MQCONNX
PO XAOPEN

Na kliencie istnieje wyjście, które może modyfikować zachowanie wywołania MQCONN lub MQCONNX nazywanego wyjściem `PreConnect` . Wyjście `PreConnect` może modyfikować dowolne parametry wywołania MQCONN lub MQCONNX, w tym nazwę menedżera kolejek. Klient najpierw wywołuje to wyjście, a następnie wywołanie MQCONN lub MQCONNX. Należy zauważyć, że tylko początkowe wywołanie MQCONN lub MQCONNX wywołuje wyjście funkcji API. Kolejne wywołania ponownego połączenia nie mają wpływu.

Środowisko wykonawcze

Ogólnie wszystkie błędy funkcji wyjścia są przekazywane z powrotem do procedury obsługi wyjścia przy użyciu pól ExitResponse i ExitResponse2 w MQAXP.

Z kolei te błędy są przekształcane w wartości MQCC_ * i MQRC_ * i przekazywane z powrotem do aplikacji w polach CompCode i Reason. Jednak wszystkie błędy napotkane w logice procedury obsługi wyjścia są przekazywane z powrotem do aplikacji jako wartości MQCC_ * i MQRC_ * w polach CompCode i Reason.

Jeśli funkcja MQ_TERM_EXIT zwraca błąd:

- Wywołanie MQDISC zostało już wykonane
- Nie ma innej możliwości kierowania produktem *po wykonaniu funkcji wyjścia MQ_TERM_EXIT produktu* (a tym samym wykonania procedury czyszczącej środowiska wykonawczego)

- Procedura czyszcząca środowiska wykonawczego wyjścia nie jest wykonywana

Nie można rozładować wyjścia, ponieważ może być ono nadal używane. Ponadto inne zarejestrowane wyjścia w dalszej kolejności w łańcuchu wyjścia, dla których wyjście *przed* powiodło się, będą sterowane w kolejności odwrotnej.

Konfigurowanie środowiska wykonawczego programów zewnętrznych

Podczas przetwarzania jawnego wywołania MQCONN lub MQCONNX logika obsługi wyjścia konfiguruje środowisko wykonawcze wyjścia przed wywołaniem funkcji inicjowania wyjścia (MQ_INIT_EXIT). Konfiguracja środowiska wykonawczego wyjścia obejmuje ładowanie wyjścia, uzyskiwanie pamięci masowej i inicjowanie struktur parametrów wyjścia. Przydzielany jest również uchwyt konfiguracji wyjścia.

Jeśli podczas tej fazy wystąpią błędy, wywołanie MQCONN lub MQCONNX zakończy się niepowodzeniem z kodem CompCode MQCC_FAILED i jednym z następujących kodów przyczyny:

BŁĄD ŁADOWANIA MQRC_API_EXIT_LOAD_ERROR
Próba załadowania modułu wyjścia funkcji API nie powiodła się.

MQRC_API_EXIT_NOT_FOUND (nie znaleziono)
Nie można znaleźć funkcji wyjścia funkcji API w module wyjścia funkcji API.

MQRC_STORAGE_NIEDOSTĘPNY
Próba zainicjowania środowiska wykonawczego dla funkcji wyjścia funkcji API nie powiodła się z powodu braku wystarczającej ilości dostępnej pamięci masowej.

MQRC_API_EXIT_INIT_BŁĄD
Wystąpił błąd podczas inicjowania środowiska wykonawczego dla funkcji wyjścia funkcji API.

Czyszczenie środowiska wykonawczego programów zewnętrznych

Podczas przetwarzania jawnego wywołania MQDISC lub niejawnego żądania rozłączenia w wyniku zakończenia aplikacji logika obsługi wyjścia może wymagać wyczyszczenia środowiska wykonawczego wyjścia po wywołaniu funkcji zakończenia wyjścia (MQ_TERM_EXIT), jeśli jest zarejestrowana.

Czyszczenie środowiska wykonawczego wyjścia obejmuje zwolnienie pamięci masowej dla struktur parametrów wyjścia, prawdopodobnie usunięcie wszystkich modułów poprzednio załadowanych do pamięci.

Jeśli podczas tej fazy wystąpią błędy, jawne wywołanie MQDISC zakończy się niepowodzeniem z kodem CompCode MQCC_FAILED i następującym kodem przyczyny (błędy nie są podświetlane w niejawnych żądaniach odłączenia):

BŁĄD MQRC_API_EXIT_TERM_ERROR
Wystąpił błąd podczas zamykania środowiska wykonawczego dla funkcji wyjścia funkcji API. Wyjście nie powinno zwracać żadnego niepowodzenia z MQDISC przed wywołaniami funkcji wyjścia API MQ_TERM* ani po tych wywołaniach.

Wyjścia funkcji API na klientach

Klient używa wyjścia PreConnect do modyfikowania zachowania wywołań MQCONN i MQCONNX i nie obsługuje właściwości wyjścia funkcji API.

Wyjście PreConnect

Na kliencie wyjście PreConnect może być używane do wyszukiwania definicji kanału w centralnym repozytorium, takim jak serwer LDAP.

Wyjście PreConnect może również modyfikować dowolny parametr lub wszystkie parametry samego wywołania MQCONN lub MQCONNX, na przykład nazwę menedżera kolejek.

W przypadku aplikacji klienckich wyjście PreConnect musi zostać wywołane przed wyjściem funkcji API, ponieważ wyjście funkcji API MQCONN lub MQCONNX jest wywoływane tylko wtedy, gdy znana jest nazwa menedżera kolejek i nazwa ta może zostać zmieniona przez wyjście funkcji PreConnect .

Należy zauważyć, że tylko początkowe wywołanie MQCONN lub MQCONNX wywołuje wyjście.

Właściwości wyjścia funkcji API

Na serwerze wyjścia funkcji API mogą rejestrować strukturę MQXEPO w czasie inicjowania. Struktura MQXEPO zawiera pole ExitProperties , w którym znajduje się grupa właściwości interesujących dla wyjścia. Powoduje to wygenerowanie oddzielnego uchwytu właściwości komunikatu, którym wyjście może manipulować niezależnie od dowolnego uchwytu właściwości komunikatu aplikacji.

Na kliencie właściwości wyjścia funkcji API nie są obsługiwane. Próba zarejestrowania nazwy grupy właściwości na kliencie zakończy się niepowodzeniem z kodem przyczyny MQRC_EXIT_PROPS_NOT_SUPPORTED.

Wycofanie-MQ_BACK_EXIT

Funkcja MQ_BACK_EXIT udostępnia funkcję wycofania wyjścia do wykonania *przed* i *po* przetwarzaniu wycofania. Użyj identyfikatora funkcji MQXF_BACK z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wycofaniu wywołań funkcji wyjścia.

Interfejs do tej funkcji to:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Początek-MQ_BEGIN_EXIT

MQ_BEGIN_EXIT udostępnia funkcję rozpoczęcia wyjścia do wykonania *przed* i *po* przetworzeniu wywołania MQBEGIN. Użyj identyfikatora funkcji MQXF_BEGIN z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQBEGIN.

Interfejs do tej funkcji to:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

Opcje pBegin(PMQBO)-wejście/wyjście
Wskaźnik do opcji początku.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn, &pBeginOptions, &CompCode,
               &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Wywołanie zwrotne-MQ_CALLBACK_EXIT

Usługa MQ_CALLBACK_EXIT udostępnia funkcję wyjściową do wykonywania przetwarzania wywołań zwrotnych *przed* i *po* . Użyj identyfikatora funkcji MQXF_CALLBACK z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia wywołania zwrotnego.

Interfejs do tej funkcji to:

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts,
                  &pBuffer, &pMQCBContext)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu

Hconn (MQHCONN)-wejście/wyjście
Uchwyt połączenia

Opis pMsg
deskryptor komunikatu

pGetMsgOpts
Opcje sterujące działaniem komendy MQGET

pBuffer
Obszar, który ma zawierać dane komunikatu

pMQCBContext
Dane kontekstu dla wywołania zwrotnego

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc, &pGetMsgOpts, &pBuffer,
               &pContext);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
void MQENTRY MQ_CALLBACK_EXIT (
PMQAXP    pExitParms;    /* Exit parameter structure */
PMQAXC    pExitContext;  /* Exit context structure */
PMQHCONN  pHconn;        /* Connection handle */
PPMQMD    ppMsgDesc;     /* Message descriptor */
PPMQGMO   ppGetMsgOpts;  /* Options that define the operation of the consumer */
PPMQVOID  ppBuffer;      /* Area to contain the message data */
PPMQCBC   ppContext;)    /* Context data for the callback */
```

Użycie notatek

1. Wyjście wywołania zwrotnego jest wywoływane przed wywołaniem konsumenta i po zakończeniu funkcji konsumenta. Mimo że struktury MQMD i MQGMO można zmieniać, zmiana wartości w wyjściu poprzedzającym nie powoduje ponownego pobrania komunikatu z kolejki, ponieważ komunikat został już usunięty z kolejki, która ma zostać dostarczona do funkcji konsumenta.

Zarządzanie funkcjami zwrotnymi-MQ_CB_EXIT

Usługa MQ_CB_EXIT udostępnia funkcję wyjścia, która wykonuje *przed* i *po* wywołaniu obiektu MQCB. Użyj identyfikatora funkcji MQXF_CB z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQCB.

Interfejs do tej funkcji to:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &pCallbackDesc,
            &Hobj, &pMsgDesc, &pGetMsgOpts, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu

Hconn (MQHCONN)-wejście/wyjście
Uchwyt połączenia

Operacja (MQLONG)-wejście/wyjście
Wartość operacji

pCallbackOpis (PMQCBD)-wejście/wyjście
Deskryptor wywołania zwrotnego

Hobj (MQHOBJ)-wejście/wyjście
Uchwyt obiektu

pMsgDesc (PMQMD)-wejście/wyjście
deskryptor komunikatu

pGetMsgOpts (PMQGMO)-wejście/wyjście
Opcje sterujące działaniem obiektu MQCB

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący się do kodu CompCode

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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. */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_CB_EXIT (&ExitParms, &ExitContext, &Hconn, &Operation, &Hobj, &pMsgDesc,
            &pGetMsgOpts, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Zamknij-MQ_CLOSE_EXIT

Usługa MQ_CLOSE_EXIT udostępnia funkcję zamykania wyjścia w celu wykonania *przed* i *po* przetworzeniu wywołania MQCLOSE. Użyj identyfikatora funkcji MQXF_CLOSE z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQCLOSE.

Interfejs do tej funkcji to:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pHobj (PMQHOBJ)-wejście
Wskaźnik do uchwytu obiektu.

Opcje (MQLONG)-wejście/wyjście
Zamknij opcje.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kodem zakończenia jest MQCC_FAILED, funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext,&Hconn, &pHobj, &Options,
               &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Zatwierdzenie-MQ_CMIT_EXIT

Usługa MQ_CMIT_EXIT udostępnia funkcję wyjścia zatwierdzania, która wykonuje *przed* i *po* zatwierdzeniu. Użyj identyfikatora funkcji MQXF_CMIT z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* zatwierdzeniu wywołania funkcji obsługi wyjścia.

Jeśli operacja zatwierdzania nie powiedzie się i transakcja zostanie wycofana, wywołanie MQCMIT zakończy się niepowodzeniem z MQCC_WARNING i MQRC_BACKED_OUT. Te kody powrotu i przyczyny są przekazywane do dowolnego *po* funkcjach wyjścia MQCMIT w celu wskazania, że jednostka pracy została wycofana.

Interfejs do tej funkcji to:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext,&Hconn, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Użycie notatek

1. Opisany tutaj interfejs funkcji MQ_GET_EXIT jest używany zarówno dla funkcji wyjścia MQXF_GET, jak i dla funkcji wyjścia “MQXF_DATA_CONV_ON_GET” na stronie 1641 .

   Dla tych dwóch funkcji wyjścia zdefiniowano osobne punkty wejścia, dlatego aby przechwycić *oba* wywołanie MQXEP musi zostać użyte dwukrotnie. W przypadku tego wywołania należy użyć identyfikatora funkcji MQXF_GET.

   Ponieważ interfejs MQ_GET_EXIT jest taki sam dla MQXF_GET i MQXF_DATA_CONV_ON_GET, w obu przypadkach można użyć pojedynczej funkcji wyjścia. Pole *Function* w strukturze MQAXP wskazuje, która funkcja wyjścia została wywołana. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla tych dwóch przypadków.

Rozszerzenie Connect i Connect-MQ_CONNX_EXIT

Usługa MQ_CONNX_EXIT udostępnia funkcję wyjścia połączenia, która wykonuje *przed* i *po* przetworzeniu MQCONN, oraz funkcję wyjścia rozszerzenia połączenia, która wykonuje *przed* i *po* przetworzeniu MQCONNX.

Ten sam interfejs, jak opisano w tej sekcji, jest wywoływany zarówno dla funkcji wyjścia wywołania MQCONN, jak i MQCONNX.

Gdy agent kanału komunikatów (MCA) odpowiada na przychodzące połączenie klienta, agent MCA może nawiązać połączenie i utworzyć pewną liczbę wywołań funkcji API IBM MQ , zanim stan klienta stanie się w pełni znany. Te funkcje API wywołują funkcje wyjścia interfejsu API za pomocą MQAXC w oparciu o sam program MCA (na przykład w polach UserId i ConnectionName interfejsu MQAXC).

Gdy agent MCA odpowiada na kolejne przychodzące wywołania interfejsu API klienta, struktura MQAXC jest oparta na kliencie przychodzącym i odpowiednio ustawia pola UserId i ConnectionName .

Nazwa menedżera kolejek ustawiona przez aplikację w wywołaniu MQCONN lub MQCONNX jest przekazywana do bazowego wywołania połączenia. Każda próba zmiany nazwy menedżera kolejek przez produkt *przed wykonaniem operacji* MQ_CONNX_EXIT nie ma wpływu na działanie menedżera kolejek.

Użyj identyfikatorów funkcji MQXF_CONN i MQXF_CONNX z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQCONN i MQCONNX.

Wyjście MQ_CONNX_EXIT wywołane z powodu wywołania funkcji MQXR_BEFORE *nie może* wywoływać żadnych wywołań funkcji API języka IBM MQ , ponieważ w tym momencie nie zostało skonfigurowane poprawne środowisko.

Usługa MQ_CONNX_EXIT nie może wywołać wywołania MQDISC z wywołania wyjścia API dla połączenia, dla którego jest wywoływana. To ograniczenie ma zastosowanie zarówno do wyjść funkcji API klienta, jak i serwera.

Interfejsy MQCONN i MQCONNX są identyczne:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOpts,
&pHconn, &CompCode, &Reason);
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

pQMgrNazwa (PMQCHAR)-wejście
Wskaźnik do nazwy menedżera kolejek podanej w wywołaniu MQCONNX. Wyjście nie może zmieniać tej nazwy w wywołaniu MQCONN lub MQCONNX.

pConnectOpts (PMQCNO)-wejście/wyjście
Wskaźnik do opcji sterujących działaniem wywołania MQCONNX.

Szczegółowe informacje można znaleźć w sekcji “MQCNO-opcje połączenia” na stronie 321.

W przypadku funkcji wyjścia MQXF_CONN opcja pConnectwskazuje domyślną strukturę opcji połączenia (MQCNO_DEFAULT).

pHconn (PMQHCONN)-wejście
Wskaźnik do uchwytu połączenia.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie)

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, &pQMgrName, &pConnectOps,
               &pHconn, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Użycie notatek

1. Opisany tutaj interfejs funkcji MQ_CONNX_EXIT jest używany zarówno dla wywołania MQCONN, jak i wywołania MQCONNX. Jednak dla tych dwóch wywołań zdefiniowano osobne punkty wejścia. Aby przechwycić *oba* wywołania, należy użyć wywołania MQXEP co najmniej dwa razy z identyfikatorem funkcji MQXF_CONN, a następnie ponownie z identyfikatorem MQXF_CONNX.

   Ponieważ interfejs MQ_CONNX_EXIT jest taki sam dla wywołań MQCONN i MQCONNX, dla obu wywołań można użyć funkcji pojedynczego wyjścia. Pole *Function* w strukturze MQAXP wskazuje, które wywołanie jest w toku. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla tych dwóch wywołań.
2. Gdy agent kanału komunikatów (MCA) odpowiada na przychodzące połączenie klienta, agent MCA może wywołać pewną liczbę wywołań produktu MQ , zanim stan klienta stanie się w pełni znany. Te wywołania produktu MQ powodują wywołanie funkcji wyjścia API ze strukturą MQAXC zawierającą dane związane z agentem MCA, a nie z klientem (na przykład identyfikatorem użytkownika i nazwą połączenia). Jeśli jednak stan klienta jest w pełni znany, kolejne wywołania produktu MQ powodują wywołanie funkcji wyjścia funkcji API z odpowiednimi danymi klienta w strukturze MQAXC.
3. Wszystkie funkcje wyjścia MQXR_BEFORE są wywoływane przed sprawdzaniem poprawności parametrów przez menedżer kolejek. Dlatego parametry mogą być niepoprawne (w tym niepoprawne wskaźniki dla adresów parametrów).

Funkcja MQ_CONNX_EXIT jest wywoływana przed sprawdzaniem autoryzacji przez menedżer kolejek.

4. Funkcja wyjścia nie może zmieniać nazwy menedżera kolejek określonego w wywołaniu MQCONN lub MQCONNX. Jeśli nazwa zostanie zmieniona przez funkcję wyjścia, wyniki będą niezdefiniowane.
5. Funkcja wyjścia MQXR_BEFORE dla usługi MQ_CONNX_EXIT nie może wykonywać wywołań produktu MQ innych niż MQXEP.

Wywołanie zwrotne sterujące-MQ_CTL_EXIT

Usługa MQ_CTL_EXIT udostępnia funkcję wyjścia żądania subskrypcji, która wykonuje przetwarzanie wywołania zwrotnego sterowania *przed* i *po* . Użyj identyfikatora funkcji MQXF_CTL z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania zwrotnego sterowania.

Interfejs do tej funkcji to:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason)
```

gdzie parametry są następujące:

Hconn (MQHCONN)-wejście/wyjście
Uchwyt połączenia.

Wejście/wyjście operacji (MQLONG)
Operacja przetwarzana w wywołaniu zwrotnym zdefiniowanym dla określonego uchwytu obiektu

Wejście/wyjście ControlOpts (MQCTLO)
Opcje sterujące działaniem komendy MQCTL

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_CTL_EXIT (&Hconn, &Operation, &ControlOpts, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Rozłączenie-MQ_DISC_EXIT

MQ_DISC_EXIT udostępnia funkcję wyjścia rozłączania w celu wykonania *przed* i *po* przetworzeniu wyjścia MQDISC. Użyj identyfikatora funkcji MQXF_DISC z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia MQDISC.

Interfejs do tej funkcji to

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
&CompCode, &Reason);
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

pHconn (PMQHCONN)-wejście
Wskaźnik do uchwytu połączenia.

*W przypadku poprzedzającego wywołania MQDISC*wartość tego pola jest jedną z następujących wartości:

- Uchwyt połączenia zwrócony w wywołaniu MQCONN lub MQCONNX
- Zero dla środowisk, w których adapter specyficzny dla środowiska nawiązał połączenie z menedżerem kolejek
- Wartość ustawiona przez poprzednie wywołanie funkcji wyjścia

*W przypadku wywołania po MQDISC*wartość tego pola to zero lub wartość ustawiona przez poprzednie wywołanie funkcji wyjścia.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
PMQHCONN        pHconn;         /* Ptr to Connection handle */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Pobieranie-MQ_GET_EXIT

Usługa MQ_GET_EXIT udostępnia funkcję wyjścia pobierania, która wykonuje *przed* i *po* przetworzeniu wywołania MQGET.

Istnieją dwa identyfikatory funkcji:

1. Użyj opcji MQXF_GET z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQGET.
2. Informacje na temat używania identyfikatora funkcji MQXF_DATA_CONV_ON_GET zawiera sekcja “MQXF_DATA_CONV_ON_GET” na stronie 1641 .

Interfejs do tej funkcji to:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

Hobj (MQHOBJ)-wejście/wyjście
Uchwyt obiektu.

pMsgDesc (PMQMD)-wejście/wyjście
Wskaźnik do deskryptora komunikatu.

pGetMsgOpts (PMQGMO)-wejście/wyjście
Wskaźnik do pobrania opcji komunikatu.

BufferLength (MQLONG)-wejście/wyjście
Długość buforu komunikatów.

pBuffer (PMQBYTE)-wejścia/wyjścia
Wskaźnik do buforu komunikatów.

pDataDługość (PMQLONG)-wejście/wyjście
Wskaźnik do pola długości danych.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pGetMsgOpts, &BufferLength, &pBuffer, &pDataLength,
             &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Użycie notatek

1. Opisany tutaj interfejs funkcji MQ_GET_EXIT jest używany zarówno dla funkcji wyjścia MQXF_GET, jak i dla funkcji wyjścia “MQXF_DATA_CONV_ON_GET” na stronie 1641 .

Dla tych dwóch funkcji wyjścia zdefiniowano osobne punkty wejścia, dlatego aby przechwycić *oba* wywołanie MQXEP musi zostać użyte dwukrotnie. W przypadku tego wywołania należy użyć identyfikatora funkcji MQXF_GET.

Ponieważ interfejs MQ_GET_EXIT jest taki sam dla MQXF_GET i MQXF_DATA_CONV_ON_GET, w obu przypadkach można użyć pojedynczej funkcji wyjścia. Pole *Function* w strukturze MQAXP wskazuje, która funkcja wyjścia została wywołana. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla tych dwóch przypadków.

MQXF_DATA_CONV_ON_GET

Identyfikator funkcji MQXF_DATA_CONV_ON_GET jest używany z usługą MQ_GET_EXIT.

Informacje o interfejsie tego wywołania i przykładowej deklaracji języka C zawiera sekcja MQ_GET_EXIT .

Użycie notatek

Po zarejestrowaniu ten punkt wejścia jest wywoływany, gdy komunikaty docierają do aplikacji, ale przed jakąkolwiek konwersją danych. Może to być przydatne, jeśli wyjście funkcji API musi wykonać przetwarzanie, takie jak deszyfrowanie lub dekompresja, przed przekazaniem komunikatu do konwersji danych. Wyjście może w razie potrzeby spowodować pominięcie konwersji danych przez zwrócenie wartości MQXCC_SUPPRESS_FUNCTION; więcej informacji na ten temat zawiera opis struktury MQAXP .

Zarejestrowanie tego punktu wejścia na kliencie powoduje, że konwersja danych jest wykonywana lokalnie na komputerze klienta. W celu poprawnego działania może być konieczne zainstalowanie na kliencie wyjść konwersji aplikacji. Należy zauważyć, że opcja MQXF_DATA_CONV_ON_GET jest również używana na potrzeby przetwarzania asynchronicznego.

Jeśli używane jest wywołanie MQ_GET_EXIT, należy użyć wywołania MQXF_DATA_CONV_ON_GET z przyczyną wyjścia MQXR_BEFORE w celu zarejestrowania *przed* funkcją wyjścia konwersji danych MQGET.

Nie ma funkcji wyjścia MQXR_AFTER dla MQXF_DATA_CONV_ON_GET; funkcja wyjścia MQXR_AFTER dla MQXF_GET zapewnia wymaganą możliwość przetwarzania wyjścia po konwersji danych.

Dla wywołania MQ_GET_EXITzdefiniowano osobne punkty wejścia, dlatego aby przechwycić *obie* funkcje wyjścia, należy dwukrotnie użyć wywołania MQXEP. W przypadku tego wywołania należy użyć identyfikatora funkcji MQXF_DATA_CONV_ON_GET.

Ponieważ interfejs MQ_GET_EXIT jest taki sam dla MQXF_GET i MQXF_DATA_CONV_ON_GET, w obu przypadkach można użyć pojedynczej funkcji wyjścia. Pole *Function* w strukturze MQAXP wskazuje, która funkcja wyjścia została wywołana. Alternatywnie można użyć wywołania MQXEP w celu zarejestrowania różnych funkcji wyjścia dla tych dwóch przypadków.

Inicjowanie-MQ_INIT_EXIT

Usługa MQ_INIT_EXIT udostępnia inicjowanie na poziomie połączenia, wskazywane przez ustawienie parametru ExitReason w produkcie MQAXP na wartość MQXR_CONNECTION.

Podczas inicjowania należy zwrócić uwagę na następujące kwestie:

- Funkcja MQ_INIT_EXIT wywołuje funkcję MQXEP w celu zarejestrowania komend API IBM MQ oraz punktów wejścia i wyjścia, w których jest ona zainteresowana.
- Wyjścia nie muszą przechwytywać wszystkich komend API IBM MQ . Funkcje wyjścia są wywoływane tylko wtedy, gdy zarejestrowano zainteresowanie.
- Pamięć, która ma być używana przez wyjście, może zostać uzyskana podczas inicjowania.
- Jeśli wywołanie tej funkcji nie powiedzie się, wywołanie MQCONN lub MQCONNX, które wywołało tę funkcję, również zakończy się niepowodzeniem z kodem CompCode i przyczyną, które zależą od wartości pola ExitResponse w MQAXP.
- Wyjście MQ_INIT_EXIT nie może wydawać wywołań funkcji API języka IBM MQ , ponieważ w tym momencie nie zostało skonfigurowane poprawne środowisko.

- Jeśli operacja MQ_INIT_EXIT zakończy się niepowodzeniem z błędem MQXCC_FAILED, menedżer kolejek zwraca dane z wywołania MQCONN lub MQCONNX, które wywołało wywołanie MQCC_FAILED i MQRC_API_EXIT_ERROR.
- Jeśli menedżer kolejek napotka błąd podczas inicjowania środowiska wykonawczego funkcji wyjścia interfejsu API przed wywołaniem pierwszego wywołania MQ_INIT_EXIT, menedżer kolejek powraca z wywołania MQCONN lub MQCONNX, które wywołało wywołanie MQ_INIT_EXIT z wywołaniami MQCC_FAILED i MQRC_API_EXIT_INIT_ERROR.

Interfejs do pliku MQ_INIT_EXIT to:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

CompCode (MQLONG)-wejście/wyjście
Wskaźnik do kodu zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Wskaźnik do kodu przyczyny kwalifikującego kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Kod CompCode i przyczyna zwracane do aplikacji zależą od wartości pola ExitResponse w MQAXP.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        MQAXP           ExitParms;      /* Exit parameter structure */
        MQAXC           ExitContext;    /* Exit context structure */
        MQLONG          CompCode;       /* Completion code */
        MQLONG          Reason;         /* Reason code */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Użycie notatek

1. Funkcja MQ_INIT_EXIT może wywołać funkcję MQXEP w celu zarejestrowania adresów funkcji wyjścia dla konkretnych wywołań produktu MQ , które mają zostać przechwycone. Nie ma potrzeby przechwytywania wszystkich wywołań produktu MQ ani przechwytywania zarówno wywołań MQXR_BEFORE, jak i MQXR_AFTER. Na przykład pakiet wyjścia może przechwycić tylko wywołanie MQPUT MQXR_BEFORE.
2. Pamięć masową, która ma być używana przez funkcje wyjścia w pakiecie wyjścia, można uzyskać za pomocą funkcji MQ_INIT_EXIT. Alternatywnie, funkcje wyjścia mogą uzyskać pamięć, gdy są wywoływane, w razie potrzeby. Jednak cała pamięć masowa powinna zostać zwolniona przed zakończeniem pakietu wyjścia; funkcja MQ_TERM_EXIT może zwolnić pamięć masową lub wywołać wcześniej funkcję wyjścia.
3. Jeśli komenda MQ_INIT_EXIT zwróci wartość MQXCC_FAILED w polu `ExitResponse` w MQAXP lub zakończy się niepowodzeniem w inny sposób, wywołanie MQCONN lub MQCONNX, które spowodowało wywołanie komendy MQ_INIT_EXIT, również nie powiedzie się, a parametry **CompCode** i **Reason** zostaną ustawione na odpowiednie wartości.
4. Funkcja MQ_INIT_EXIT nie może wykonywać wywołań produktu MQ innych niż MQXEP.

Zapytanie-MQ_INQ_EXIT

MQ_INQ_EXIT udostępnia funkcję wyjścia zapytania do wykonania *przed* i *po* przetworzeniu wywołania MQINQ. Użyj identyfikatora funkcji MQXF_INQ z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQINQ.

Interfejs do tej funkcji to:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

Hobj (MQHOBJ)-wejście
Uchwyt obiektu.

SelectorCount (MQLONG)-wejście
Liczba selektorów

pSelectors (PMQLONG)-wejście/wyjście
Wskaźnik do tablicy wartości selektora.

IntAttrLiczba (MQLONG)-dane wejściowe
Liczba atrybutów całkowitych.

pIntAttrs (PMQLONG)-wejście/wyjście
Wskaźnik do tablicy wartości atrybutów całkowitych.

CharAttrDługość (MQLONG)-wejście/wyjście
Długość tablicy atrybutów znakowych.

pCharAttrs (PMQCHAR)-wejście/wyjście
Wskaźnik do tablicy atrybutów znakowych.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Otwarcie-MQ_OPEN_EXIT

Usługa MQ_OPEN_EXIT udostępnia funkcję otwartego wyjścia do wykonywania operacji *przed* i *po* przetworzeniu wywołania MQOPEN. Użyj identyfikatora funkcji MQXF_OPEN z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQOPEN.

Interfejs do tej funkcji to

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
&pHobj, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pObjDesc (PMQOD)-wejście/wyjście
Wskaźnik do deskryptora obiektu.

Opcje (MQLONG)-wejście/wyjście
Otwórz opcje.

pHobj (PMQHOBJ)-wejście
Wskaźnik do uchwytu obiektu.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
              &pHobj, &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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

Usługa MQ_PUT_EXIT udostępnia funkcję wyjścia umieszczania w celu wykonania *przed* i *po* przetworzeniu wywołania MQPUT. Użyj identyfikatora funkcji MQXF_PUT z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQPUT.

Interfejs do tej funkcji to:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

Hobj (MQHOBJ)-wejście/wyjście
Uchwyt obiektu.

pMsgDesc (PMQMD)-wejście/wyjście
Wskaźnik do deskryptora komunikatu.

pPutMsgOpts (PMQPMO)-wejście/wyjście
Wskaźnik do opcji umieszczania komunikatów.

BufferLength (MQLONG)-wejście/wyjście
Długość buforu komunikatów.

pBuffer (PMQBYTE)-wejścia/wyjścia
Wskaźnik do buforu komunikatów.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &pMsgDesc,
             &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Użycie notatek

- Komunikaty raportu wygenerowane przez menedżera kolejek pomijają normalne przetwarzanie wywołania. Z tego powodu takie komunikaty nie mogą być przechwytywane przez funkcję MQ_PUT_EXIT ani funkcję MQPUT1 . Jednak komunikaty raportów wygenerowane przez agenta kanału komunikatów są przetwarzane normalnie i mogą zostać przechwycone przez funkcję MQ_PUT_EXIT lub MQ_PUT1_EXIT . Aby przechwycić wszystkie komunikaty raportu wygenerowane przez agenta MCA, należy użyć zarówno MQ_PUT_EXIT, jak i MQ_PUT1_EXIT .

Put1 - MQ_PUT1_EXIT

Klasa MQ_PUT1_EXIT udostępnia funkcję wyjścia *umieszczania tylko jednego komunikatu* na potrzeby wykonywania operacji *przed* i *po przetworzeniu wywołania* MQPUT1 . Użyj identyfikatora funkcji MQXF_PUT1 z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* MQPUT1 wywołaj funkcje wyjścia.

Interfejs do tej funkcji to:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
&pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pObjDesc (PMQOD)-wejście/wyjście
Wskaźnik do deskryptora obiektu.

pMsgDesc (PMQMD)-wejście/wyjście
Wskaźnik do deskryptora komunikatu.

pPutMsgOpts (PMQPMO)-wejście/wyjście
Wskaźnik do opcji umieszczania komunikatów.

BufferLength (MQLONG)-wejście/wyjście
Długość buforu komunikatów.

pBuffer (PMQBYTE)-wejścia/wyjścia
Wskaźnik do buforu komunikatów.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &pMsgDesc,
              &pPutMsgOpts, &BufferLength, &pBuffer, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Zestaw-MQ_SET_EXIT

Usługa MQ_SET_EXIT udostępnia funkcję wyjścia do wykonywania *przed* i *po* przetwarzaniu wywołania MQSET. Użyj identyfikatora funkcji MQXF_SET z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania MQSET.

Interfejs do tej funkcji to:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttr, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

Hobj (MQHOBJ)-wejście
Uchwyt obiektu.

SelectorCount (MQLONG)-wejście
Liczba selektorów

pSelectors (PMQLONG)-wejście/wyjście
Wskaźnik do tablicy wartości selektora.

IntAttrLiczba (MQLONG)-dane wejściowe
Liczba atrybutów całkowitych.

pIntAttrs (PMQLONG)-wejście/wyjście
Wskaźnik do tablicy wartości atrybutów całkowitych.

CharAttrDługość (MQLONG)-wejście/wyjście
Długość tablicy atrybutów znakowych.

pCharAttrs (PMQCHAR)-wejście/wyjście
Wskaźnik do wartości atrybutów znakowych.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj, &SelectorCount,
             &pSelectors, &IntAttrCount, &pIntAttrs, &CharAttrLength,
             &pCharAttrs, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Status-MQ_STAT_EXIT

Usługa MQ_STAT_EXIT udostępnia funkcję wyjścia statusu do wykonywania *przed* i *po* przetworzeniu wywołania MQSTAT. Użyj identyfikatora funkcji MQXF_STAT z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniu funkcji wyjścia MQSTAT.

Interfejs do tej funkcji to:

```
MQ_STAT_EXIT (&ExitParms, &ExitContext, &Hconn, &Type, &pStatus
              &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

Typ (MQLONG)-wejście
Typ informacji o statusie do pobrania.

pStatus (PMQSTS)-dane wyjściowe
Wskaźnik do buforu statusu.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Zakończenie-MQ_TERM_EXIT

MQ_TERM_EXIT zapewnia zakończenie na poziomie połączenia, zarejestrowane z identyfikatorem funkcji MQXF_TERM i ExitReason MQXR_CONNECTION. Po zarejestrowaniu usługa MQ_TERM_EXIT jest wywoływana raz dla każdego żądania rozłączenia.

W ramach zakończenia można zwolnić pamięć masową, która nie jest już wymagana przez wyjście, a także można wykonać wszystkie wymagane procedury czyszczące.

Jeśli operacja MQ_TERM_EXIT nie powiedzie się z błędem MQXCC_FAILED, menedżer kolejek zwraca z usługi MQDISC, która wywołała tę usługę, komunikaty MQCC_FAILED i MQRC_API_EXIT_ERROR.

Jeśli menedżer kolejek napotka błąd podczas kończenia działania środowiska wykonawczego funkcji wyjścia API po wywołaniu ostatniej usługi MQ_TERM_EXIT, menedżer kolejek powraca z wywołania MQDISC, które wywołało usługę MQ_TERM_EXIT z wywołaniami MQCC_FAILED i MQRC_API_EXIT_TERM_ERROR.

Interfejs do tej funkcji to:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kodem zakończenia jest MQCC_FAILED, funkcja wyjścia może ustawić pole kodu przyczyny na dowolną poprawną wartość MQRC_ *.

Kod CompCode i przyczyna zwracane do aplikacji zależą od wartości pola ExitResponse w MQAXP.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
MQAXP           ExitParms;      /* Exit parameter structure */
MQAXC           ExitContext;    /* Exit context structure */
MQLONG          CompCode;       /* Completion code */
MQLONG          Reason;         /* Reason code */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode, &Reason)
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Użycie notatek

1. Funkcja MQ_TERM_EXIT jest opcjonalna. Nie jest konieczne, aby pakiet wyjścia zarejestrował wyjście zakończenia, jeśli nie ma konieczności przetwarzania zakończenia.

   Jeśli funkcje należące do pakietu obsługi wyjścia uzyskują zasoby podczas połączenia, funkcja MQ_TERM_EXIT jest wygodnym miejscem, w którym można zwolnić te zasoby, na przykład zwalniać pamięć masową uzyskaną dynamicznie.
2. Jeśli funkcja MQ_TERM_EXIT jest rejestrowana podczas wywoływania wywołania MQDISC, funkcja wyjścia jest wywoływana po wywołaniu wszystkich funkcji wyjścia MQDISC.
3. Jeśli wartość MQ_TERM_EXIT zwraca wartość MQXCC_FAILED w polu `ExitResponse` w MQAXP lub nie powiedzie się w inny sposób, wywołanie MQDISC, które spowodowało wywołanie MQ_TERM_EXIT, również kończy się niepowodzeniem z parametrami **CompCode** i **Reason** ustawionymi na odpowiednie wartości.

Rejestrowanie subskrypcji-MQ_SUB_EXIT

MQ_SUB_EXIT udostępnia funkcję wyjściową do wykonywania *przed* i *po* przetwarzaniu ponownej rejestracji subskrypcji. Użyj identyfikatora funkcji MQXF_SUB z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* funkcjach wyjścia wywołania rejestracji subskrypcji.

Interfejs do tej funkcji to:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub, &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście/wyjście
Uchwyt połączenia.

pSubDesc-input/output
Tablica selektorów atrybutów.

pHobj -wejście/wyjście
Uchwyt obiektu

pHsub (MQHOBJ) wejścia/wyjścia
Uchwyt subskrypcji

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pSubDesc, &pHobj, &pHsub,
             &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Żądanie subskrypcji-MQ_SUBRQ_EXIT

Usługa MQ_SUBRQ_EXIT udostępnia funkcję wyjścia żądania subskrypcji, która umożliwia wykonywanie operacji *przed* i *po* przetworzeniu żądania subskrypcji. Użyj identyfikatora funkcji MQXF_SUBRQ z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować *przed* i *po* wywołaniach funkcji obsługi wyjścia żądań subskrypcji.

Interfejs do tej funkcji to:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście/wyjście
Uchwyt połączenia.

pHsub (MQHOBJ) wejścia/wyjścia
Uchwyt subskrypcji

Wejście/wyjście działania (MQLONG)
Działanie

pSubRqOpts (MQSRO) wejście/wyjście

CompCode (MQLONG)-wejście/wyjście
Kod zakończenia, poprawne wartości to:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się

Przyczyna (MQLONG)-wejście/wyjście
Kod przyczyny kwalifikujący kod zakończenia.

Jeśli kod zakończenia to MQCC_OK, jedyną poprawną wartością jest:

MQRC_BRAK
(0, x '000') Brak powodu do zgłaszania.

Jeśli kod zakończenia to MQCC_FAILED lub MQCC_WARNING, funkcja wyjścia może ustawić w polu kodu przyczyny dowolną poprawną wartość MQRC_ *.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
        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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
               &CompCode, &Reason);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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

Funkcja XA_CLOSE_EXIT udostępnia funkcję wyjścia xa_close, która ma być wykonywana przed i po przetwarzaniu xa_close. Użyj identyfikatora funkcji MQXF_XACLOSE z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_close.

Interfejs do tej funkcji to:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXa_info (PMQCHAR)-wejście/wyjście
Informacje o menedżerze zasobów specyficzne dla instancji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjściową xa_commit do wykonania przed i po przetwarzaniu xa_commit. Użyj identyfikatora funkcji MQXF_XACOMMIT z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_commit.

Interfejs do tej funkcji to:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_COMMIT_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjścia xa_complete do wykonania przed i po przetwarzaniu xa_complete. Użyj identyfikatora funkcji MQXF_XACOMPLETE z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_complete.

Interfejs do tej funkcji to:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pHandle (PMQLONG)-wejście/wyjście
Wskaźnik do operacji asynchronicznej.

pRetVal (PMQLONG)-wejście/wyjście
Wartość zwracana operacji asynchronicznej.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_COMPLETE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHandle, &pRetval, &Rmid, &Flags,
&XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjścia xa_end do wykonania przed i po przetwarzaniu xa_end. Użyj identyfikatora funkcji MQXF_XAEND z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po xa_end.

Interfejs do tej funkcji to:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_END_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji C:

```
typedef void MQENTRY XA_END_EXIT (
     PMQAXP   pExitParms,   /* Address of exit parameter structure */
     PMQAXC   pExitContext, /* Address of exit context structure */
     PMQHCONN pHconn,       /* Address of connection handle */
     PMQPTR   ppXID,        /* Address of transaction branch ID */
     PMQLONG  pRmid,        /* Address of resource manager identifier */
     PMQLONG  pFlags,       /* Address of resource manager options*/
     PMQLONG  pXARetCode);  /* Address of response from XA call */
```

xa_forget-XA_FORGET_EXIT

XA_FORGET_EXIT udostępnia funkcję wyjścia xa_forget do wykonania przed i po przetwarzaniu xa_forget. Użyj identyfikatora funkcji MQXF_XAFORGET z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_forget.

Interfejs do tej funkcji to:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_FORGET_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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

Funkcja XA_OPEN_EXIT udostępnia funkcję wyjścia xa_open, która ma być wykonywana przed i po przetwarzaniu xa_open. Użyj identyfikatora funkcji MQXF_XAOPEN z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_open.

Interfejs do tej funkcji to:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXa_info (PMQCHAR)-wejście/wyjście
Informacje o menedżerze zasobów specyficzne dla instancji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn, &pXa_info, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjścia xa_prepare do wykonania przed i po przetwarzaniu xa_prepare. Użyj identyfikatora funkcji MQXF_XAPREPARE z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_prepare.

Interfejs do tej funkcji to:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_PREPARE_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję xa_recover do wykonania przed i po przetwarzaniu xa_recover. Użyj identyfikatora funkcji MQXF_XARECOVER z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_recover.

Interfejs do tej funkcji to:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Liczba (MQLONG)-wejście/wyjście
Maksymalna liczba XID w tablicy XID

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_RECOVER_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Count, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjściową xa_rollback do wykonania przed i po przetwarzaniu xa_rollback. Użyj identyfikatora funkcji MQXF_XAROLLBACK z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_rollback.

Interfejs do tej funkcji to:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_ROLLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjścia xa_start do wykonania przed i po przetwarzaniu xa_start. Użyj identyfikatora funkcji MQXF_XASTART z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po operacji xa_start.

Interfejs do tej funkcji to:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
XA_START_EXIT (&ExitParms, &ExitContext, &Hconn, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjściową ax_reg, która ma być wykonywana przed i po przetwarzaniu ax_reg. Użyj identyfikatora funkcji MQXF_AXREG z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po ax_reg.

Interfejs do tej funkcji to:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode)
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Hconn (MQHCONN)-wejście
Uchwyt połączenia.

pXID (MQPTR)-wejście/wyjście
Identyfikator gałęzi transakcji.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
AX_REG_EXIT (&ExitParms, &ExitContext, &pXID, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 udostępnia funkcję wyjściową ax_unreg, która ma być wykonywana przed i po przetworzeniu ax_unreg. Użyj identyfikatora funkcji MQXF_AXUNREG z przyczynami wyjścia MQXR_BEFORE i MQXR_AFTER, aby zarejestrować funkcje wyjścia wywołania przed i po ax_unreg.

Interfejs do tej funkcji to:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

gdzie parametry są następujące:

ExitParms (MQAXP)-wejście/wyjście
Struktura parametru wyjścia.

ExitContext (MQAXC)-wejście/wyjście
Wyjdź ze struktury kontekstu.

Rmid (MQLONG)-wejście/wyjście
Identyfikator menedżera zasobów.

Flagi (MQLONG)-wejście/wyjście
Opcje menedżera zasobów.

XARetCode (MQLONG)-wejście/wyjście
Odpowiedź z wywołania XA.

Wywołanie w języku C

Menedżer kolejek logicznie definiuje następujące zmienne:

```
     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 */
```

Następnie menedżer kolejek wywołuje logicznie wyjście w następujący sposób:

```
AX_UNREG_EXIT (&ExitParms, &ExitContext, &Rmid, &Flags, &XARetCode);
```

Wyjście musi być zgodne z następującym prototypem funkcji 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 */
```

Informacje ogólne o wywoływaniu funkcji wyjścia

Ten temat zawiera ogólne wskazówki dotyczące planowania wyjść, w szczególności dotyczące obsługi błędów i nieoczekiwanych zdarzeń.

Niepowodzenie wyjścia

Jeśli funkcja wyjścia zostanie nieprawidłowo zakończona po destrukcyjnym, poza punktem synchronizacji, wywołaniu MQGET, ale przed przekazaniem komunikatu do aplikacji, procedura obsługi wyjścia może odzyskać sprawność po awarii i przekazać sterowanie do aplikacji.

W takim przypadku komunikat może zostać utracony. Jest tak, jak w przypadku awarii aplikacji natychmiast po odebraniu komunikatu z kolejki.

Wywołanie MQGET może zakończyć się z MQCC_FAILED i MQRC_API_EXIT_ERROR.

Jeśli funkcja wyjścia wywołania funkcji API *przed* zakończy działanie nieprawidłowo, procedura obsługi wyjścia może odzyskać sprawność po awarii i przekazać sterowanie do aplikacji bez przetwarzania wywołania funkcji API. W takim przypadku funkcja wyjścia musi odzyskać wszystkie zasoby, których jest właścicielem.

Jeśli używane są połączone wyjścia, można sterować *po* wyjściach wywołania funkcji API dla dowolnego *przed* wyjściami wywołania funkcji API, które zostały pomyślnie sterowane. Wywołanie funkcji API może zakończyć się niepowodzeniem z wywołaniami MQCC_FAILED i MQRC_API_EXIT_ERROR.

Przykładowa obsługa błędów dla funkcji obsługi wyjścia

Poniższy diagram przedstawia punkty (e *N*) w którym mogą wystąpić błędy. Jest to tylko przykład pokazujący zachowanie wyjść i powinien być odczytywany razem z poniższą tabelą. W tym przykładzie dwie funkcje wyjścia są wywoływane zarówno przed, jak i po każdym wywołaniu funkcji API w celu pokazania zachowania z łańcuchowymi wyjściami.

```
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
```

Poniższa tabela zawiera listę działań, które należy wykonać w każdym punkcie błędu. Pokryto tylko podzbiór punktów błędów, ponieważ przedstawione tutaj reguły mogą mieć zastosowanie do wszystkich innych. To działania określają zamierzone zachowanie w każdym przypadku.

Tabela 837. Błędy wyjścia funkcji API i odpowiednie działania do wykonania

Err Pt	**Opis**	**Działania**
e1	Błąd podczas konfigurowania środowiska.	1. Cofnij konfigurację środowiska zgodnie z wymaganiami 2. Brak funkcji wyjścia napędu 3. Niepowodzenie MQCONN z MQCC_FAILED, MQRC_API_EXIT_LOAD_ERROR
e2	Funkcja MQ_INIT_EXIT kończy działanie z następującymi danymi: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Wyczyść środowisko 2. Niepowodzenie MQCONN z MQCC_FAILED, MQRC_API_EXIT_INIT_ERROR • Dla MQXCC_ * 1. Działa jak dla wartości MQXCC_ * i MQXR2_ * [1] 2. Wyczyść środowisko

Tabela 837. Błędy wyjścia funkcji API i odpowiednie działania do wykonania (kontynuacja)		
Err Pt	**Opis**	**Działania**
e3	*Przed zakończeniem działania funkcji* MQ_CONNX_EXIT 1: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Drive MQ_TERM_EXIT function (Funkcja MQ_TERM_EXIT) 2. Wyczyść środowisko 3. Niepowodzenie wywołania MQCONN z MQCC_FAILED, MQRC_API_EXIT_ERROR • Dla MQXCC_ * 1. Działa jak dla wartości MQXCC_ * i MQXR2_ * [1] 2. Funkcja MQ_TERM_EXIT napędu, jeśli jest wymagana 3. Czyszczenie środowiska, jeśli jest wymagane
e4	*Przed zakończeniem działania funkcji* MQ_CONNX_EXIT 2: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Dysk *po wykonaniu funkcji* MQ_CONNX_EXIT 1 2. Drive MQ_TERM_EXIT function (Funkcja MQ_TERM_EXIT) 3. Wyczyść środowisko 4. Niepowodzenie wywołania MQCONN z MQCC_FAILED, MQRC_API_EXIT_ERROR • Dla MQXCC_ * 1. Działa jak dla wartości MQXCC_ * i MQXR2_ * [1] 2. Napęd *po funkcji* MQ_CONNX_EXIT 1, jeśli wyjście nie jest pomijane 3. Funkcja MQ_TERM_EXIT napędu, jeśli jest wymagana 4. Czyszczenie środowiska, jeśli jest wymagane
e5	Wywołanie MQCONN nie powiodło się.	1. Przekaż MQCONN CompCode i przyczynę 2. Napęd *po wykonaniu funkcji* MQ_CONNX_EXIT 2, jeśli *przed* pomyślnie zakończono działanie MQ_CONNX_EXIT 2, a wyjście nie jest pomijane 3. Napęd *po wykonaniu funkcji* MQ_CONNX_EXIT 1, jeśli *przed* pomyślnie zakończono działanie MQ_CONNX_EXIT 1, a wyjście nie jest pomijane 4. Drive MQ_TERM_EXIT function (Funkcja MQ_TERM_EXIT) 5. Wyczyść środowisko
e6	*Po zakończeniu działania funkcji* MQ_CONNX_EXIT 2: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Dysk *po wykonaniu funkcji* MQ_CONNX_EXIT 1 2. Zakończ wywołanie MQCONN z MQCC_FAILED, MQRC_API_EXIT_ERROR • Dla MQXCC_ * 1. Działa jak dla wartości MQXCC_ * i MQXR2_ * [1] 2. Napęd *po wykonaniu funkcji* MQ_CONNX_EXIT 1, jeśli jest to wymagane

Tabela 837. Błędy wyjścia funkcji API i odpowiednie działania do wykonania (kontynuacja)		
Err Pt	**Opis**	**Działania**
e7	*Po* Funkcja MQ_CONNX_EXIT 1 kończy działanie z następującymi danymi: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED, zakończ wywołanie MQCONN z MQCC_FAILED, MQRC_API_EXIT_ERROR • W przypadku MQXCC_ * należy działać jak w przypadku wartości MQXCC_ * i MQXR2_ * [1]
e8	*Przed zakończeniem działania funkcji* MQ_OPEN_EXIT 1: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED, zakończ wywołanie MQOPEN z MQCC_FAILED, MQRC_API_EXIT_ERROR • W przypadku MQXCC_ * należy działać jak w przypadku wartości MQXCC_ * i MQXR2_ * [1]
e9	*Przed zakończeniem działania funkcji* MQ_OPEN_EXIT 2 z następującymi danymi: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Dysk *po wykonaniu funkcji* MQ_OPEN_EXIT 1 2. Zakończenie wywołania MQOPEN z MQCC_FAILED, MQRC_API_EXIT_ERROR • W przypadku MQXCC_ * należy działać jak w przypadku wartości MQXCC_ * i MQXR2_ * [1]
e10	Wywołanie MQOPEN nie powiodło się	1. Przekaż kod MQOPEN CompCode i przyczynę 2. Napęd *po funkcji* MQ_OPEN_EXIT 2, jeśli wyjście nie jest pomijane 3. Dysk *po wykonaniu funkcji* MQ_OPEN_EXIT 1, jeśli wyjście nie jest pomijane, a wyjście łańcuchowe nie jest pomijane
e11	*Po* Funkcja MQ_OPEN_EXIT 2 kończy działanie z następującymi danymi: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Dysk *po wykonaniu funkcji* MQ_OPEN_EXIT 1 2. Zakończenie wywołania MQOPEN z MQCC_FAILED, MQRC_API_EXIT_ERROR • Dla MQXCC_ * 1. Działa jak dla wartości MQXCC_ * i MQXR2_ * [1] 2. Napęd *po funkcji* MQ_OPEN_EXIT 1, jeśli wyjście nie jest pomijane
e25	*Po zakończeniu działania funkcji* MQ_DISC_EXIT 2: • MQXCC-NIEPOWODZENIE • MQXCC_ *	• Dla MQXCC_FAILED: 1. Dysk *po wykonaniu funkcji* MQ_DISC_EXIT 1 2. Drive MQ_TERM_EXIT function (Funkcja MQ_TERM_EXIT) 3. Wyczyść środowisko wykonawcze wyjścia 4. Zakończ wywołanie MQDISC z MQCC_FAILED, MQRC_API_EXIT_ERROR • Dla MQXCC_ * 1. Działa jak dla wartości MQXCC_ * i MQXR2_ * [1] 2. Drive MQ_TERM_EXIT function (Funkcja MQ_TERM_EXIT) 3. Wyczyść środowisko wykonawcze wyjścia

Uwaga:

1. Wartości MQXCC_ * i MQXR2_* oraz odpowiadające im działania są zdefiniowane w sekcji W jaki sposób menedżery kolejek przetwarzają funkcje wyjścia.

Pola ExitResponse są ustawione niepoprawnie

Ta sekcja zawiera informacje o tym, co może się stać, gdy w polu ExitResponse zostanie ustawiona dowolna wartość poza obsługiwanymi wartościami.

Jeśli w polu ExitResponse jest ustawiona wartość inna niż jedna z obsługiwanych wartości, mają zastosowanie następujące czynności:

- Dla *przed* funkcją wyjścia funkcji API MQCONN lub MQDISC:
  - Wartość ExitResponse2 jest ignorowana.
  - Nie są wywoływane dalsze funkcje wyjścia *przed* w łańcuchu wyjścia (jeśli istnieją); samo wywołanie funkcji API nie jest wykonywane.
  - W przypadku wyjść *przed* , które zostały pomyślnie wywołane, wyjścia *po* są wywoływane w odwrotnej kolejności.
  - Jeśli zostały zarejestrowane, funkcje wyjścia zakończenia dla tych *przed* funkcjami wyjścia MQCONN lub MQDISC w łańcuchu, które zostały pomyślnie wywołane, są sterowane w celu wykonania procedury czyszczącej po wykonaniu tych funkcji wyjścia.
  - Wywołanie MQCONN lub MQDISC kończy się niepowodzeniem z błędem MQRC_API_EXIT_ERROR.
- Dla funkcji wyjścia funkcji API *przed* IBM MQ innej niż MQCONN lub MQDISC:
  - Wartość ExitResponse2 jest ignorowana.
  - W łańcuchu wyjścia (jeśli istnieją) nie są wywoływane dalsze funkcje konwersji danych *przed* lub *po* .
  - W przypadku wyjść *przed* , które zostały pomyślnie wywołane, wyjścia *po* są wywoływane w odwrotnej kolejności.
  - Samo wywołanie funkcji API IBM MQ nie jest wysyłane.
  - Wywołanie funkcji API IBM MQ kończy się niepowodzeniem z błędem MQRC_API_EXIT_ERROR.
- W przypadku funkcji wyjścia funkcji API *po* MQCONN lub MQDISC:
  - Wartość ExitResponse2 jest ignorowana.
  - Pozostałe funkcje wyjścia, które zostały pomyślnie wywołane przed wywołaniem funkcji API, są wywoływane w odwrotnej kolejności.
  - Po zarejestrowaniu funkcje wyjścia przerwania dla tych *przed* lub *po* funkcjach wyjścia MQCONN lub MQDISC w łańcuchu, które zostały pomyślnie wywołane, są sterowane w celu wykonania procedury czyszczącej po wyjściu.
  - Kod CompCode o wartości poważniejszej niż MQCC_WARNING i CompCode zwrócony przez wyjście jest zwracany do aplikacji.
  - Do aplikacji zwracana jest przyczyna błędu MQRC_API_EXIT_ERROR.
  - Wywołanie funkcji API IBM MQ zostało pomyślnie wykonane.
- W przypadku funkcji wyjścia wywołania funkcji API *after* IBM MQ innej niż MQCONN lub MQDISC:
  - Wartość ExitResponse2 jest ignorowana.
  - Pozostałe funkcje wyjścia, które zostały pomyślnie wywołane przed wywołaniem funkcji API, są wywoływane w odwrotnej kolejności.
  - Kod CompCode o wartości poważniejszej niż MQCC_WARNING i CompCode zwrócony przez wyjście jest zwracany do aplikacji.
  - Do aplikacji zwracana jest przyczyna błędu MQRC_API_EXIT_ERROR.
  - Wywołanie funkcji API IBM MQ zostało pomyślnie wykonane.
- Dla funkcji konwersji danych *przed* w przypadku funkcji get exit:
  - Wartość ExitResponse2 jest ignorowana.

- Pozostałe funkcje wyjścia, które zostały pomyślnie wywołane przed wywołaniem funkcji API, są wywoływane w odwrotnej kolejności.
- Komunikat nie jest przekształcany, a nieprzekształcony komunikat jest zwracany do aplikacji.
- Kod CompCode o wartości poważniejszej niż MQCC_WARNING i CompCode zwrócony przez wyjście jest zwracany do aplikacji.
- Do aplikacji zwracana jest przyczyna błędu MQRC_API_EXIT_ERROR.
- Wywołanie funkcji API IBM MQ zostało pomyślnie wykonane.

Uwaga: Ponieważ błąd dotyczy wyjścia, lepiej jest zwrócić MQRC_API_EXIT_ERROR niż MQRC_NOT_CONVERTED.

Jeśli funkcja wyjścia ustawi w polu ExitResponse2 wartość inną niż jedna z obsługiwanych wartości, zostanie przyjęta wartość MQXR2_DEFAULT_CONTINUATION .

Informacje uzupełniające o interfejsie usług instalowalnych

Ta kolekcja tematów zawiera informacje uzupełniające na temat usług instalowalnych.

Funkcje i typy danych są wymienione w porządku alfabetycznym w grupie dla każdego typu usługi.

Pojęcia pokrewne

ALW Instalowalne usługi i komponenty dla systemów UNIX, Linux i Windows

IBM i Instalowalne usługi i komponenty systemu IBM i

IBM i Informacje uzupełniające o interfejsie instalowalnych usług dla systemu IBM i

Zadania pokrewne

Rozszerzanie narzędzi menedżera kolejek

ALW Konfigurowanie instalowalnych usług

Sposób wyświetlania funkcji

Sposób, w jaki opisano funkcje instalowalnych usług.

Dla każdej funkcji istnieje opis, w tym identyfikator funkcji (dla MQZEP).

Parametry są wyświetlane w kolejności, w jakiej muszą wystąpić. Wszystkie one muszą być obecne.

Po każdej nazwie parametru występuje jego typ danych. Są to podstawowe typy danych opisane w sekcji “Podstawowe typy danych” na stronie 236.

Po opisie parametrów podano również wywołanie w języku C.

MQZ_AUTHENTICATE_USER-Uwierzytelnianie użytkownika

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest wywoływana przez menedżer kolejek w celu uwierzytelnienia użytkownika lub ustawienia pól kontekstu tożsamości. Jest on wywoływany po ustanowieniu kontekstu aplikacji użytkownika IBM MQ .

Kontekst aplikacji jest ustanawiany podczas wywołań połączenia w punkcie, w którym kontekst użytkownika aplikacji jest inicjowany, i w każdym punkcie, w którym kontekst użytkownika aplikacji jest zmieniany. Za każdym razem, gdy wykonywane jest wywołanie połączenia, informacje o kontekście użytkownika aplikacji są ponownie uzyskiwane w polu *IdentityContext* .

Identyfikator tej funkcji (dla MQZEP) to MQZID_AUTHENTICATE_USER.

Składnia

MQZ_AUTHENTICATE_USER (*QMgrName* , *SecurityParms* , *ApplicationContext* , *IdentityContext* , *CorrelationPtr* , *ComponentData* , *Continuation* , *CompCode* , *Przyczyna*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

SecurityParms
Typ: MQCSP-input

Parametry zabezpieczeń. Dane dotyczące identyfikatora użytkownika, hasła i typu uwierzytelniania. Jeśli atrybut AuthenticationType struktury MQCSP jest określony jako MQCSP_AUTH_USER_ID_AND_PWD, zarówno identyfikator użytkownika, jak i hasło są porównywane z odpowiednimi polami w parametrze IdentityContext (MQZIC) w celu określenia, czy są zgodne. Więcej informacji na ten temat zawiera “MQCSP-parametry zabezpieczeń” na stronie 341.

Podczas wywołania MQCONN MQI ten parametr zawiera wartość NULL lub wartości domyślne.

ApplicationContext
Typ: MQZAC-wejście

Kontekst aplikacji. Dane dotyczące aplikacji wywołującej. Szczegółowe informacje na ten temat zawiera sekcja MQZAC-Kontekst aplikacji .

Podczas każdego wywołania MQCONN lub MQCONNX MQI ponownie uzyskiwane są informacje o kontekście użytkownika w strukturze MQZAC.

IdentityContext
Typ: MQZIC-wejście/wyjście

Kontekst tożsamości. Na wejściu do funkcji uwierzytelniania użytkownika identyfikuje ona bieżący kontekst tożsamości. Funkcja uwierzytelniania użytkownika może to zmienić. W tym momencie menedżer kolejek przyjmuje nowy kontekst tożsamości. Więcej informacji na temat struktury MQZIC zawiera sekcja MQZIC-Identity context (Kontekst tożsamości MQZIC).

CorrelationPtr
Typ: MQPTR-wyjście

Wskaźnik korelacji. Określa adres wszystkich danych korelacji. Wskaźnik ten jest następnie przekazywany do innych wywołań OAM.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDatawywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Flaga kontynuacji. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależna od innych składników.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Zadeklaruj parametry przekazywane do usługi w następujący sposób:

```
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-Sprawdzanie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i uruchamiana przez menedżer kolejek w celu sprawdzenia, czy jednostka ma uprawnienia do wykonania określonego działania lub działań na określonym obiekcie.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_CHECK_AUTHORITY.

Składnia

MQZ_CHECK_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityName
Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa jednostki, której uprawnienia do obiektu mają zostać sprawdzone. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

Nie jest konieczne, aby ta jednostka była znana bazowej usłudze zabezpieczeń. Jeśli nie jest znana, do sprawdzenia używane są autoryzacje specjalnej grupy **nobody** (nikt), do której należą wszystkie obiekty. Pusta nazwa jest poprawna i może być użyta w ten sposób.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez EntityName. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

Uprawnienie
Typ: MQLONG-input

Uprawnienie do sprawdzenia. Jeśli sprawdzana jest jedna autoryzacja, to pole jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli sprawdzana jest więcej niż jedna autoryzacja, jest to bitowe LUB odpowiednich stałych MQZAO_ *.

Do korzystania z wywołań MQI mają zastosowanie następujące autoryzacje:

MQZAO_CONNECT
Możliwość użycia wywołania MQCONN.

MQZAO_BROWSE,
Możliwość użycia wywołania MQGET z opcją przeglądania.

Umożliwia to określenie opcji MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_BROWSE_NEXT w wywołaniu MQGET.

MQZAO_INPUT
Podmiot kerberos. Możliwość użycia wywołania MQGET z opcją wejściową.

Umożliwia to określenie opcji MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE lub MQOO_INPUT_AS_Q_DEF w wywołaniu MQOPEN.

MQZAO_WYNIK
Możliwość użycia wywołania MQPUT.

Umożliwia to określenie opcji MQOO_OUTPUT w wywołaniu MQOPEN.

MQZAO_INQUIRE,
Możliwość użycia wywołania MQINQ.

Umożliwia to określenie opcji MQOO_INQUIRE w wywołaniu MQOPEN.

MQZAO_SET
Możliwość użycia wywołania MQSET.

Umożliwia to określenie opcji MQOO_SET w wywołaniu MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT,
Możliwość przekazywania kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_PASS_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_IDENTITY_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_PASS_ALL_CONTEXT (mqzao_pass_all)
Możliwość przekazania całego kontekstu.

Umożliwia to określenie opcji MQOO_PASS_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT,
Możliwość ustawienia kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_SET_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_SET_IDENTITY_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_ALL_CONTEXT (mqzao_set_all)
Możliwość ustawienia całego kontekstu.

Umożliwia to określenie opcji MQOO_SET_ALL_CONTEXT w wywołaniu MQOPEN i opcji MQPMO_SET_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY (uprawnienie użytkownika na przemian)
Możliwość korzystania z alternatywnych uprawnień użytkownika.

Umożliwia to określenie opcji MQOO_ALTERNATE_USER_AUTHORITY w wywołaniu MQOPEN, a opcji MQPMO_ALTERNATE_USER_AUTHORITY w wywołaniu MQPUT1 .

MQZAO_ALL_MQI
Wszystkie autoryzacje MQI.

Spowoduje to włączenie wszystkich autoryzacji.

Do administrowania menedżerem kolejek mają zastosowanie następujące autoryzacje:

MQZAO_CREATE
Możliwość tworzenia obiektów określonego typu.

MQZAO_USUŃ
Możliwość usunięcia określonego obiektu.

MQZAO_DISPLAY
Możliwość wyświetlania atrybutów określonego obiektu.

MQZAO_CHANGE
Możliwość zmiany atrybutów określonego obiektu.

MQZAO_CLEAR,
Możliwość usuwania wszystkich komunikatów z określonej kolejki.

Autoryzacja MQZAO_AUTORYZACJI
Możliwość autoryzowania innych użytkowników dla określonego obiektu.

MQZAO_CONTROL
Możliwość uruchamiania lub zatrzymywania obiektu nasłuchiwania, usługi lub kanału innego niż kanał klienta oraz możliwość wysyłania pakietów ping do obiektu kanału innego niż kanał klienta.

MQZAO_CONTROL_EXTENDED,
Możliwość zresetowania numeru kolejnego lub rozstrzygnięcia wątpliwego komunikatu w obiekcie kanału innego niż kanał klienta.

MQZAO_ALL_ADMIN
Możliwość ustawienia kontekstu tożsamości.

Wszystkie autoryzacje administracyjne inne niż MQZAO_CREATE.

Następujące autoryzacje mają zastosowanie zarówno do używania interfejsu MQI, jak i do administrowania menedżerem kolejek:

MQZAO_WSZYSTKIE
Wszystkie autoryzacje inne niż MQZAO_CREATE.

MQZAO_BRAK
Brak autoryzacji.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

Jeśli wywołanie komponentu nie powiedzie się (*CompCode* zwraca wartość MQCC_FAILED), a parametr *Continuation* ma wartość MQZCI_DEFAULT lub MQZCI_CONTINUE, menedżer kolejek kontynuuje wywoływanie innych komponentów (jeśli istnieją).

Jeśli wywołanie powiedzie się (czyli kod *CompCode* zwróci wartość MQCC_OK), żadne inne komponenty nie zostaną wywołane bez względu na ustawienie opcji *Kontynuacja* .

Jeśli wywołanie nie powiedzie się, a parametr *Continuation* ma wartość MQZCI_STOP, nie zostaną wywołane żadne inne komponenty, a błąd zostanie zwrócony do menedżera kolejek. Komponenty nie mają informacji o poprzednich wywołaniach, dlatego parametr *Kontynuacja* jest zawsze ustawiany na wartość MQZCI_DEFAULT przed wywołaniem.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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 -uprawnienie do sprawdzania (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i uruchamiana przez menedżer kolejek w celu sprawdzenia, czy jednostka ma uprawnienia do wykonania określonego działania lub działań na określonym obiekcie.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_CHECK_AUTHORITY.

Komenda MQZ_CHECK_AUTHORITY_2 jest podobna do komendy MQZ_CHECK_AUTHORITY, ale z parametrem **EntityName** zastąpionym przez parametr **EntityData** .

Składnia

MQZ_CHECK_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityData
Typ: MQZED-wejście

Dane jednostki. Dane dotyczące jednostki z uprawnieniami do obiektu, który ma zostać sprawdzony. Szczegółowe informacje można znaleźć w sekcji "MQZED-deskryptor jednostki" na stronie 1730.

Nie jest konieczne, aby ta jednostka była znana bazowej usłudze zabezpieczeń. Jeśli nie jest znana, do sprawdzenia używane są autoryzacje specjalnej grupy **nobody** (nikt), do której należą wszystkie obiekty. Pusta nazwa jest poprawna i może być użyta w ten sposób.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Uprawnienie do sprawdzenia. Jeśli sprawdzana jest jedna autoryzacja, to pole jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli sprawdzana jest więcej niż jedna autoryzacja, jest to bitowe LUB odpowiednich stałych MQZAO_ *.

Do korzystania z wywołań MQI mają zastosowanie następujące autoryzacje:

MQZAO_CONNECT
Możliwość użycia wywołania MQCONN.

MQZAO_BROWSE,
Możliwość użycia wywołania MQGET z opcją przeglądania.

Umożliwia to określenie opcji MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_BROWSE_NEXT w wywołaniu MQGET.

MQZAO_INPUT
Podmiot kerberos. Możliwość użycia wywołania MQGET z opcją wejściową.

Umożliwia to określenie opcji MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE lub MQOO_INPUT_AS_Q_DEF w wywołaniu MQOPEN.

MQZAO_WYNIK
Możliwość użycia wywołania MQPUT.

Umożliwia to określenie opcji MQOO_OUTPUT w wywołaniu MQOPEN.

MQZAO_INQUIRE,
Możliwość użycia wywołania MQINQ.

Umożliwia to określenie opcji MQOO_INQUIRE w wywołaniu MQOPEN.

MQZAO_SET
Możliwość użycia wywołania MQSET.

Umożliwia to określenie opcji MQOO_SET w wywołaniu MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT,
Możliwość przekazywania kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_PASS_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_IDENTITY_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_PASS_ALL_CONTEXT (mqzao_pass_all)
Możliwość przekazania całego kontekstu.

Umożliwia to określenie opcji MQOO_PASS_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT,
Możliwość ustawienia kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_SET_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_SET_IDENTITY_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_ALL_CONTEXT (mqzao_set_all)
Możliwość ustawienia całego kontekstu.

Umożliwia to określenie opcji MQOO_SET_ALL_CONTEXT w wywołaniu MQOPEN i opcji MQPMO_SET_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY (uprawnienie użytkownika na przemian)
Możliwość korzystania z alternatywnych uprawnień użytkownika.

Umożliwia to określenie opcji MQOO_ALTERNATE_USER_AUTHORITY w wywołaniu MQOPEN, a opcji MQPMO_ALTERNATE_USER_AUTHORITY w wywołaniu MQPUT1 .

MQZAO_ALL_MQI
Wszystkie autoryzacje MQI.

Spowoduje to włączenie wszystkich autoryzacji.

Do administrowania menedżerem kolejek mają zastosowanie następujące autoryzacje:

MQZAO_CREATE
Możliwość tworzenia obiektów określonego typu.

MQZAO_USUŃ
Możliwość usunięcia określonego obiektu.

MQZAO_DISPLAY
Możliwość wyświetlania atrybutów określonego obiektu.

MQZAO_CHANGE
Możliwość zmiany atrybutów określonego obiektu.

MQZAO_CLEAR,
Możliwość usuwania wszystkich komunikatów z określonej kolejki.

Autoryzacja MQZAO_AUTORYZACJI
Możliwość autoryzowania innych użytkowników dla określonego obiektu.

MQZAO_CONTROL
Możliwość uruchamiania lub zatrzymywania obiektu nasłuchiwania, usługi lub kanału innego niż kanał klienta oraz możliwość wysyłania pakietów ping do obiektu kanału innego niż kanał klienta.

MQZAO_CONTROL_EXTENDED,
Możliwość zresetowania numeru kolejnego lub rozstrzygnięcia wątpliwego komunikatu w obiekcie kanału innego niż kanał klienta.

MQZAO_ALL_ADMIN
Możliwość ustawienia kontekstu tożsamości.

Wszystkie autoryzacje administracyjne inne niż MQZAO_CREATE.

Następujące autoryzacje mają zastosowanie zarówno do używania interfejsu MQI, jak i do administrowania menedżerem kolejek:

MQZAO_WSZYSTKIE
Wszystkie autoryzacje inne niż MQZAO_CREATE.

MQZAO_BRAK
Brak autoryzacji.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                       ObjectName, ObjectType, Authority, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-Sprawdź, czy użytkownik jest uprzywilejowany

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_6 i jest wywoływana przez menedżer kolejek w celu określenia, czy określony użytkownik jest użytkownikiem uprzywilejowanym.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_CHECK_PRIVILEGED.

Składnia

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityData
Typ: MQZED-wejście

Dane jednostki. Dane dotyczące jednostki, która ma zostać sprawdzona. Więcej informacji na ten temat zawiera sekcja "MQZED-deskryptor jednostki" na stronie 1730.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez EntityData. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ComponentData
Typ: MQBYTExComponentDataLength -wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

Jeśli wywołanie komponentu nie powiedzie się (*CompCode* zwraca wartość MQCC_FAILED), a parametr *Continuation* ma wartość MQZCI_DEFAULT lub MQZCI_CONTINUE, menedżer kolejek kontynuuje wywoływanie innych komponentów (jeśli istnieją).

Jeśli wywołanie powiedzie się (czyli kod *CompCode* zwróci wartość MQCC_OK), żadne inne komponenty nie zostaną wywołane bez względu na ustawienie opcji *Kontynuacja* .

Jeśli wywołanie nie powiedzie się, a parametr *Continuation* ma wartość MQZCI_STOP, nie zostaną wywołane żadne inne komponenty, a błąd zostanie zwrócony do menedżera kolejek. Komponenty nie mają informacji o poprzednich wywołaniach, dlatego parametr *Kontynuacja* jest zawsze ustawiany na wartość MQZCI_DEFAULT przed wywołaniem.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_PRIVILEGED
(2584, X'A18') Ten użytkownik nie jest użytkownikiem uprzywilejowanym.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-kopiowanie wszystkich uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji. Jest on uruchamiany przez menedżer kolejek w celu skopiowania wszystkich autoryzacji, które aktualnie obowiązują dla obiektu odniesienia, do innego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_COPY_ALL_AUTHORITY.

Składnia

MQZ_COPY_ALL_AUTHORITY(*QMgrName* , *RefObjectName* , *ObjectName* , *ObjectType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

RefObjectNazwa
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu odniesienia. Nazwa obiektu odniesienia, którego autoryzacje mają zostać skopiowane. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego mają zostać ustawione dostępy. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *RefObjectName* i *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT

.

TEMAT_MQOT

.

ComponentData

Typ: MQBYTExComponentDataLength -wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDatawywołania MQZ_INIT_AUTHORITY.

Kontynuacja

Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode

Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Nieznany obiekt odniesienia.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-uprawnienie do usuwania

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i uruchamiana przez menedżer kolejek w celu usunięcia wszystkich autoryzacji powiązanych z określonym obiektem.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_DELETE_AUTHORITY.

Składnia

MQZ_DELETE_AUTHORITY(*QMgrName* , *ObjectName* , *ObjectType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego mają zostać usunięte dostępy. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDatawywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_ENUMERATE_AUTHORITY_DATA-wyliczone dane uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_4 i jest wielokrotnie uruchamiana przez menedżer kolejek w celu pobrania wszystkich danych uprawnień zgodnych z kryteriami wyboru określonymi w pierwszym wywołaniu.

Identyfikator tej funkcji (dla MQZEP) to MQZID_ENUMERATE_AUTHORITY_DATA.

Składnia

MQZ_ENUMERATE_AUTHORITY_DATA(*QMgrName* , *StartEnumeration* , *Filter* , *AuthorityBufferLength* , *AuthorityBuffer* , *AuthorityDataLength* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

StartEnumeration

Typ: MQLONG-input

Flaga wskazująca, czy wywołanie może rozpocząć wyliczanie. Wskazuje, czy wywołanie może rozpocząć wyliczanie danych uprawnień, czy kontynuować wyliczanie danych uprawnień rozpoczęte poprzednim wywołaniem MQZ_ENUMERATE_AUTHORITY_DATA. Wartość jest jedną z następujących wartości:

MQZSE_START
Rozpocznij wyliczanie. Wywołanie jest uruchamiane z tą wartością, aby rozpocząć wyliczanie danych uprawnień. Parametr **Filter** określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez to i kolejne wywołania.

MQZSE_CONTINUE
Kontynuuj wyliczanie. Wywołanie jest uruchamiane z tą wartością, aby kontynuować wyliczanie danych uprawnień. W tym przypadku parametr **Filter** jest ignorowany i może zostać określony jako wskaźnik pusty (kryteria wyboru są określone przez parametr **Filter** określony przez wywołanie, które miało parametr *StartEnumeration* ustawiony na wartość MQZSE_START).

Filtr
Typ: MQZAD-wejście

Filtr. Jeśli parametr *StartEnumeration* ma wartość MQZSE_START, parametr *Filter* określa kryteria wyboru, które mają zostać użyte do wybrania zwracanych danych uprawnień. Jeśli *Filter* jest wskaźnikiem pustym, nie są używane żadne kryteria wyboru, tzn. zwracane są wszystkie dane uprawnień. Szczegółowe informacje na temat kryteriów wyboru, które mogą być używane, zawiera sekcja “MQZAD-dane uprawnień” na stronie 1727 .

Jeśli parametr *StartEnumeration* ma wartość MQZSE_CONTINUE, parametr *Filter* jest ignorowany i można go określić jako wskaźnik pusty.

Długość AuthorityBuffer
Typ: MQLONG-input

Długość *AuthorityBuffer*. Jest to długość parametru **AuthorityBuffer** w bajtach. Bufor uprawnień musi być wystarczająco duży, aby pomieścić zwracane dane.

AuthorityBuffer
Typ: MQZAD-wyjście

Dane uprawnień. Jest to bufor, w którym zwracane są dane uprawnień. Bufor musi być wystarczająco duży, aby pomieścić strukturę MQZAD, strukturę MQZED oraz najdłuższą nazwę jednostki i najdłuższą zdefiniowaną nazwę domeny.

Uwaga: Uwaga: Ten parametr jest zdefiniowany jako MQZAD, ponieważ MQZAD zawsze występuje na początku buforu. Jeśli jednak bufor jest zadeklarowany jako MQZAD, będzie on zbyt mały-musi być większy niż MQZAD, aby mógł pomieścić nazwy MQZAD, MQZED oraz nazwy jednostek i domen.

AuthorityDataDługość
Typ: MQLONG-wyjście

Długość danych zwracanych w *AuthorityBuffer*. Jeśli bufor uprawnień jest zbyt mały, parametr *AuthorityDataLength* jest ustawiony na wymaganą długość buforu, a wywołanie zwraca kod zakończenia MQCC_FAILED i kod przyczyny MQRC_BUFFER_LENGTH_ERROR.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDatawywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku opcji MQZ_ENUMERATE_AUTHORITY_DATA ma to taki sam efekt, jak w przypadku opcji MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') Brak dostępnych danych.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-wolny użytkownik

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i uruchamiana przez menedżer kolejek w celu zwolnienia przydzielonego zasobu.

Jest ona uruchamiana po zakończeniu działania aplikacji we wszystkich kontekstach użytkownika, na przykład podczas wywołania MQI MQDISC.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_FREE_USER.

Składnia

MQZ_FREE_USER(*QMgrName* , *FreeParms* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

FreeParms
Typ: MQZFP-wejście

Wolne parametry. Struktura zawierająca dane dotyczące zasobu, który ma zostać zwolniony. Szczegółowe informacje można znaleźć w sekcji “MQZFP-parametry wolne” na stronie 1732.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDatawywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Flaga kontynuacji. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależna od innych składników.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-pobieranie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnienia, które jednostka ma do uzyskania dostępu do określonego obiektu, w tym (jeśli jednostka jest elementem głównym) uprawnień posiadanych przez grupy, do których należy element główny. Uprawnienia z profili ogólnych są uwzględniane w zwracanym zestawie uprawnień.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_GET_AUTHORITY.

Składnia

MQZ_GET_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityName
Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa jednostki, której dostęp do obiektu ma zostać pobrany. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityName*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego ma zostać uzyskany dostęp. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Organ podmiotu. Jeśli jednostka ma jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ma więcej niż jedno uprawnienie, to pole jest bitową OR odpowiednich stałych MQZAO_ *.

ComponentData
Typ: MQBYTE ×ComponentDataDługość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to taki sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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 -uprawnienie do pobierania (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnień, które jednostka ma w celu uzyskania dostępu do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_GET_AUTHORITY.

Komenda MQZ_GET_AUTHORITY_2 jest podobna do komendy MQZ_GET_AUTHORITY, ale z parametrem **EntityName** zastąpionym przez parametr **EntityData** .

Składnia

MQZ_GET_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityData
Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do jednostki, dla której ma zostać pobrana autoryzacja do obiektu. Szczegółowe informacje można znaleźć w sekcji “MQZED-deskryptor jednostki” na stronie 1730.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać odtworzone uprawnienie jednostki. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Organ podmiotu. Jeśli jednostka ma jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ma więcej niż jedno uprawnienie, to pole jest bitową OR odpowiednich stałych MQZAO_ *.

ComponentData
Typ: MQBYTE ×ComponentDataDługość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, &Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-pobranie jawnego uprawnienia

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu pobrania uprawnienia, które jednostka ma do uzyskania dostępu do określonego obiektu, w tym (jeśli jednostka jest elementem głównym) uprawnień posiadanych przez grupy, do których należy element główny. Uprawnienia z profili ogólnych są uwzględniane w zwracanym zestawie uprawnień.

W systemie AIX and Linux, dla wbudowanego menedżera uprawnień do obiektów IBM MQ (OAM), zwracane jest uprawnienie, które jest posiadane tylko przez grupę podstawową użytkownika.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_GET_EXPLICIT_AUTHORITY.

Składnia

MQZ_GET_EXPLICIT_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityName
Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa jednostki, dla której ma zostać pobrany dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityName*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać odtworzone uprawnienie jednostki. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Organ podmiotu. Jeśli jednostka ma jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ma więcej niż jedno uprawnienie, to pole jest bitową OR odpowiednich stałych MQZAO_ *.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to taki sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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 -uzyskiwanie jawnych uprawnień (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i uruchamiana przez menedżer kolejek w celu pobrania uprawnienia, które grupa nazwana ma do dostępu do określonego obiektu (ale bez dodatkowych uprawnień grupy **nobody** (nikt)) lub uprawnienia, które grupa podstawowa określonej nazwy użytkownika ma do uzyskania dostępu do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_GET_EXPLICIT_AUTHORITY.

Komenda MQZ_GET_EXPLICIT_AUTHORITY_2 jest podobna do komendy MQZ_GET_EXPLICIT_AUTHORITY, ale z parametrem **EntityName** zastąpionym przez parametr **EntityData** .

Składnia

MQZ_GET_EXPLICIT_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityData
Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do jednostki, której uprawnienia do obiektu mają zostać pobrane. Szczegółowe informacje można znaleźć w sekcji “MQZED-deskryptor jednostki” na stronie 1730.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać odtworzone uprawnienie jednostki. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Organ podmiotu. Jeśli jednostka ma jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ma więcej niż jedno uprawnienie, to pole jest bitową OR odpowiednich stałych MQZAO_ *.

ComponentData
Typ: MQBYTE ×ComponentDataDługość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                              ObjectName, ObjectType, &Authority,
                              ComponentData, &Continuation,
                              &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-inicjowanie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i uruchamiana przez menedżer kolejek podczas konfigurowania komponentu. Oczekuje się wywołania MQZEP w celu udostępnienia informacji menedżerowi kolejek.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INIT_AUTHORITY.

Składnia

MQZ_INIT_AUTHORITY(*Hconfig* , *Options* , *QMgrName* , *ComponentDataLength* , *ComponentData* , *Version* , *CompCode* , *Reason*)

Parametry

Hconfig
Typ: MQHCONFIG-input

Uchwyt konfiguracji. Ten uchwyt reprezentuje konkretny inicjowany komponent. Jest on używany przez komponent podczas wywoływania menedżera kolejek za pomocą funkcji MQZEP.

Opcje
Typ: MQLONG-input

Opcje inicjowania. Musi to być jedna z następujących wartości:

Podstawowy MQZIO_PRIMARY
Inicjowanie podstawowe.

MQZIO_SECONDARY
Inicjowanie dodatkowe.

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

Długość ComponentData
Typ: MQLONG-input

Długość danych komponentu. Długość w bajtach obszaru *ComponentData* . Ta długość jest zdefiniowana w danych konfiguracyjnych komponentu.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Jest ona inicjowana zerami przed wywołaniem podstawowej funkcji inicjowania komponentu. Te dane są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w nim przez dowolną funkcję (w tym funkcję inicjowania) udostępnioną przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Wersja
Typ: MQLONG-input/output

Numer wersji. W danych wejściowych funkcji inicjowania identyfikuje najwyższy numer wersji obsługiwany przez menedżer kolejek. Funkcja inicjowania musi zmienić tę wersję, jeśli jest to konieczne, na obsługiwaną przez nią wersję interfejsu. Jeśli po zwróceniu menedżer kolejek nie obsługuje wersji zwróconej przez komponent, wywołuje funkcję MQZ_TERM_AUTHORITY komponentu i nie używa więcej tego komponentu.

Obsługiwane są następujące wartości:

MQZAS_VERSION_1
Wersja 1.

MQZAS_VERSION_2
Wersja 2.

MQZAS_VERSION_3
Wersja 3.

MQZAS_VERSION_4
Wersja 4.

MQZAS_VERSION_5
Wersja 5.

MQZAS_VERSION_6
IBM WebSphere MQ 6.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

Niepowodzenie MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Inicjowanie nie powiodło się z niezdefiniowanej przyczyny.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-zapytanie o usługę autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest uruchamiana przez menedżer kolejek w celu wykonania zapytania o obsługiwaną funkcję.

Jeśli używanych jest wiele komponentów usług, są one wywoływane w kolejności odwrotnej do kolejności, w jakiej zostały zainstalowane.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INQUIRE.

Składnia

MQZ_INQUIRE(*QMgrName* , *SelectorCount* , *Selectors* , *IntAttrCount* , *IntAttrs* , *CharAttrLength* , *CharAttrs* , *SelectorReturned* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

SelectorCount
Typ: MQLONG-input

Liczba selektorów. Liczba selektorów podanych w parametrze **Selectors** .

Wartość musi być z zakresu od 0 do 256.

Selektory
Typ: MQLONGxSelectorCount-input

Tablica selektorów. Każdy selektor identyfikuje wymagany atrybut i musi być jednym z następujących:

- MQIACF_INTERFACE_VERSION (liczba całkowita)
- MQIACF_USER_ID_SUPPORT (liczba całkowita)
- MQCACF_SERVICE_COMPONENT (znak)

Selektory można określić w dowolnej kolejności. Liczba selektorów w tablicy jest wskazywana przez parametr **SelectorCount** .

Atrybuty całkowitoliczbowe identyfikowane przez selektory są zwracane w parametrze **IntAttrs** w tej samej kolejności, w jakiej występują w produkcie *Selectors*.

Atrybuty znakowe identyfikowane przez selektory są zwracane w parametrze **CharAttrs** w tej samej kolejności, w jakiej występują w *Selectors*.

IntAttrLiczba
Typ: MQLONG-input

Liczba atrybutów całkowitych podana w parametrze IntAttrs .

Wartość musi być z zakresu od 0 do 256.

IntAttrs
Typ: MQLONG x IntAttrLiczba-dane wyjściowe

Atrybuty całkowitoliczbowe. Tablica atrybutów całkowitych. Atrybuty całkowitoliczbowe są zwracane w tej samej kolejności, co odpowiadające im selektory liczb całkowitych w tablicy *Selectors* .

Liczba CharAttr
Typ: MQLONG-input

Długość buforu atrybutów znakowych. Długość parametru **CharAttrs** w bajtach.

Wartość musi być co najmniej sumą długości żądanych atrybutów znakowych. Jeśli nie są żądane żadne atrybuty znakowe, zero jest poprawną wartością.

CharAttrs
Typ: MQLONG x CharAttr-liczba-dane wyjściowe

Bufor atrybutów znakowych. Bufor zawierający atrybuty znakowe, skonkatenowany ze sobą. Atrybuty znakowe są zwracane w tej samej kolejności, co odpowiadające im selektory znakowe w tablicy *Selectors* .

Długość buforu jest określona przez parametr liczby CharAttr.

SelectorReturned
Typ: MQLONG x SelectorCount -dane wejściowe

Selektor został zwrócony. Tablica wartości identyfikująca, które atrybuty zostały zwrócone z zestawu żądanego przez selektory w parametrze Selektory. Liczba wartości w tej tablicy jest wskazywana przez parametr **SelectorCount** . Każda wartość w tablicy jest powiązana z selektorem z odpowiedniej pozycji w tablicy selektorów. Każda wartość jest jedną z następujących:

MQZSL_XX_ENCODE_CASE_ONE zwrócona
Atrybut żądany przez odpowiedni selektor w parametrze **Selectors** został zwrócony.

MQZSL_NIE_ZWRÓCONO
Atrybut żądany przez odpowiedni selektor w parametrze **Selectors** nie został zwrócony.

Tablica jest inicjowana ze wszystkimi wartościami *MQZSL_NOT_RETURNED*. Gdy komponent usługi autoryzacji zwraca atrybut, ustawia odpowiednią wartość w tablicy na *MQZSL_NOT_RETURNED*. Umożliwia to innym komponentom usługi autoryzacji, do których jest wykonywane wywołanie zapytania, identyfikowanie, które atrybuty zostały już zwrócone.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna

Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
Za mało miejsca na atrybuty znakowe.

MQRC_INT_COUNT_TOO_SMALL
Brak wystarczającej ilości miejsca dla atrybutów całkowitoliczbowych.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SELECTOR_COUNT_ERROR
Liczba selektorów jest niepoprawna.

BŁĄD WYWOŁANIA MQRC_SELECTOR_ERROR
Selektor atrybutu jest niepoprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED
Określono zbyt wiele selektorów.

BŁĄD MQRC_INT_ATTR_COUNT_ERROR
Niepoprawna liczba atrybutów całkowitych.

BŁĄD TABELI MQRC_INT_ATTRS_ARRAY_ERROR
Niepoprawna tablica atrybutów całkowitoliczbowych.

BŁĄD MQRC_CHAR_ATTR_LENGTH_ERROR
Liczba atrybutów znakowych jest niepoprawna.

BŁĄD MQRC_CHAR_ATTRS_ERROR
Łańcuch atrybutów znakowych jest niepoprawny.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-Odśwież wszystkie autoryzacje

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_3 i jest wywoływana przez menedżer kolejek w celu odświeżenia listy autoryzacji przechowywanych wewnętrznie przez komponent.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_REFRESH_CACHE (8L).

Składnia

MQZ_REFRESH_CACHE(*QMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

ComponentData
Typ: MQBYTE ×ComponentDataDługość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak dla MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Wywołanie C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
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-ustawianie uprawnień

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest uruchamiana przez menedżer kolejek w celu ustawienia uprawnienia dostępu jednostki do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_SET_AUTHORITY.

Uwaga: Ta funkcja nadpisuje wszystkie istniejące uprawnienia. Aby zachować istniejące uprawnienia, należy je ponownie ustawić za pomocą tej funkcji.

Składnia

MQZ_SET_AUTHORITY(*QMgrName* , *EntityName* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityName
Typ: MQCHAR12 -dane wejściowe

Nazwa jednostki. Nazwa jednostki, dla której ma zostać pobrany dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityName*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Organ podmiotu. Jeśli ustawione jest jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ustawiono więcej niż jedno uprawnienie, to pole jest bitową wartością OR odpowiednich stałych MQZAO_ *.

ComponentDatarname>
Typ: MQBYTExComponentDataLength -wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to taki sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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 -uprawnienie do ustawiania (rozszerzone)

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_2 i jest uruchamiana przez menedżer kolejek w celu ustawienia uprawnienia dostępu jednostki do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_SET_AUTHORITY.

Uwaga: Ta funkcja nadpisuje wszystkie istniejące uprawnienia. Aby zachować istniejące uprawnienia, należy je ponownie ustawić za pomocą tej funkcji.

Komenda MQZ_SET_AUTHORITY_2 jest podobna do komendy MQZ_SET_AUTHORITY, ale z parametrem **EntityName** zastąpionym przez parametr **EntityData** .

Składnia

MQZ_SET_AUTHORITY_2(*QMgrName* , *EntityData* , *EntityType* , *ObjectName* , *ObjectType* , *Authority* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityData
Typ: MQZED-wejście

Dane jednostki. Dane odnoszące się do jednostki, której uprawnienia do obiektu mają zostać ustawione. Szczegółowe informacje można znaleźć w sekcji "MQZED-deskryptor jednostki" na stronie 1730.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez *EntityData*. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName
Typ: MQCHAR48 -dane wejściowe

Nazwa obiektu. Nazwa obiektu, dla którego ma zostać ustawione uprawnienie jednostki. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType
Typ: MQLONG-input

Typ obiektu. Typ jednostki określony przez *ObjectName*. Musi to być jedna z następujących wartości:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

TEMAT_MQOT
.

Uprawnienie
Typ: MQLONG-input

Organ podmiotu. Jeśli ustawione jest jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ustawiono więcej niż jedno uprawnienie, to pole jest bitową wartością OR odpowiednich stałych MQZAO_ *.

ComponentData
Typ: MQBYTE ×ComponentDataDługość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-Zakończenie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i uruchamiana przez menedżer kolejek, gdy nie wymaga już usług tego komponentu. Funkcja musi wykonać wszystkie procedury czyszczące wymagane przez komponent.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_TERM_AUTHORITY.

Składnia

MQZ_TERM_AUTHORITY(*Hconfig* , *Options* , *QMgrName* , *ComponentData* , *CompCode* , *Reason*)

Parametry

Hconfig
Typ: MQHCONFIG-input

Uchwyt konfiguracji. Ten uchwyt reprezentuje konkretny komponent, który jest przerywany. Jest on używany przez komponent podczas wywoływania menedżera kolejek za pomocą funkcji MQZEP.

Opcje
Typ: MQLONG-input

Opcje zakończenia. Musi to być jedna z następujących wartości:

MQZTO_PRIMARY
Zakończenie podstawowe.

MQZTO_SECONDARY
Zakończenie drugorzędne.

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDataw wywołaniu MQZ_INIT_AUTHORITY.

Po zakończeniu wywołania MQZ_TERM_AUTHORITY menedżer kolejek usuwa te dane.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Niepowodzenie MQRC_TERMINATION_FAILED
(2287, X'8FF') Zakończenie nie powiodło się z niezdefiniowanej przyczyny.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-nazwa usuwania

Ta funkcja jest udostępniana przez komponent usługi nazw i uruchamiana przez menedżer kolejek w celu usunięcia pozycji dla określonej kolejki.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_DELETE_NAME.

Składnia

MQZ_DELETE_NAME(*QMgrName* , *QName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

Nazwa QName
Typ: MQCHAR48 -dane wejściowe

Nazwa kolejki. Nazwa kolejki, dla której ma zostać usunięta pozycja. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze długości ComponentDataw wywołaniu MQZ_INIT_NAME.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Musi to być jedna z następujących wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

W przypadku komendy **MQZ_DELETE_NAME** menedżer kolejek nie podejmuje próby uruchomienia innego komponentu, niezależnie od tego, co jest zwracane w parametrze **Continuation** .

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Ostrzeżenie (częściowe zakończenie).

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_UNKNOWN_NAME
(2288, X'8F0') Nie znaleziono nazwy kolejki.

Uwaga: Zwrócenie tego kodu może nie być możliwe, jeśli usługa bazowa odpowie pomyślnie w tym przypadku.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,
                 &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-inicjowanie usługi nazw

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek podczas konfigurowania komponentu. Oczekuje się wywołania MQZEP w celu udostępnienia informacji menedżerowi kolejek.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INIT_NAME.

Składnia

MQZ_INIT_NAME(*Hconfig* , *Options* , *QMgrName* , *ComponentDataLength* , *ComponentData* , *Version* , *CompCode* , *Reason*)

Parametry

Hconfig
Typ: MQHCONFIG-input

Uchwyt konfiguracji. Ten uchwyt reprezentuje konkretny inicjowany komponent. Jest on używany przez komponent podczas wywoływania menedżera kolejek za pomocą funkcji MQZEP.

Opcje
Typ: MQLONG-input

Opcje inicjowania. Musi to być jedna z następujących wartości:

Podstawowy MQZIO_PRIMARY
Inicjowanie podstawowe.

MQZIO_SECONDARY
Inicjowanie dodatkowe.

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

Długość ComponentData
Typ: MQLONG-input

Długość danych komponentu. Długość w bajtach obszaru *ComponentData* . Ta długość jest zdefiniowana w danych konfiguracyjnych komponentu.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Jest ona inicjowana zerami przed wywołaniem podstawowej funkcji inicjowania komponentu. Te dane są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w nim przez dowolną funkcję (w tym funkcję inicjowania) udostępnioną przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Wersja
Typ: MQLONG-input/output

Numer wersji. W danych wejściowych funkcji inicjowania identyfikuje najwyższy numer wersji obsługiwany przez menedżer kolejek. Funkcja inicjowania musi zmienić tę wersję, jeśli jest to konieczne, na obsługiwaną przez nią wersję interfejsu. Jeśli po zwróceniu menedżer kolejek nie obsługuje wersji zwróconej przez komponent, wywołuje funkcję MQZ_TERM_NAME komponentu i nie używa więcej tego komponentu.

Obsługiwane są następujące wartości:

MQZAS_VERSION_1
Wersja 1.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

Niepowodzenie MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Inicjowanie nie powiodło się z niezdefiniowanej przyczyny.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,
               ComponentData, &Version, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-nazwa wstawiania

Ta funkcja jest udostępniana przez komponent usługi nazw i uruchamiana przez menedżer kolejek w celu wstawienia pozycji dla określonej kolejki zawierającej nazwę menedżera kolejek będącego właścicielem kolejki. Jeśli kolejka jest już zdefiniowana w usłudze, wywołanie nie powiedzie się.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INSERT_NAME.

Składnia

MQZ_INSERT_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

Nazwa QName
Typ: MQCHAR48 -dane wejściowe

Nazwa kolejki. Nazwa kolejki, dla której ma zostać wstawiona pozycja. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

ResolvedQMgrNazwa
Typ: MQCHAR48 -dane wejściowe

Rozstrzygnięta nazwa menedżera kolejek. Nazwa menedżera kolejek, na który jest tłumaczona kolejka. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

ComponentData
Typ: MQBYTE ×ComponentDataDługość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w nim przez dowolną funkcję (w tym funkcję inicjowania) udostępnioną przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_NAME.

Kontynuacja
Typ: MQLONG-input/output

Wskaźnik kontynuacji ustawiony przez komponent. W przypadku MQZ_INSERT_NAME menedżer kolejek nie podejmuje próby uruchomienia innego komponentu, niezależnie od tego, co jest zwracane w parametrze **Continuation** .

Obsługiwane są następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS
(2290, X'8F2') Obiekt kolejki już istnieje.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-nazwa wyszukiwania

Ta funkcja jest udostępniana przez komponent usługi nazw i uruchamiana przez menedżer kolejek w celu pobrania nazwy menedżera kolejek będącego właścicielem dla określonej kolejki.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_LOOKUP_NAME.

Składnia

MQZ_LOOKUP_NAME(*QMgrName* , *QName* , *ResolvedQMgrName* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName

Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

Nazwa QName

Typ: MQCHAR48 -dane wejściowe

Nazwa kolejki. Nazwa kolejki, dla której ma zostać rozstrzygnięta pozycja. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

ResolvedQMgrNazwa

Typ: MQCHAR48 -dane wyjściowe

Rozstrzygnięta nazwa menedżera kolejek. Jeśli funkcja zakończy się pomyślnie, jest to nazwa menedżera kolejek, który jest właścicielem kolejki.

Nazwa zwracana przez komponent usługi musi być dopełniona odstępami po prawej stronie do pełnej długości parametru. Nazwa nie może być zakończona znakiem o kodzie zero ani zawierać odstępów początkowych lub wewnętrznych.

ComponentData

Typ: MQBYTExComponentDataLength -wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w nim przez dowolną funkcję (w tym funkcję inicjowania) udostępnioną przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_NAME.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Dla MQZ_LOOKUP_NAME menedżer kolejek określa, czy uruchomić inny komponent usługi nazw, w następujący sposób:

- Jeśli *CompCode* ma wartość MQCC_OK, nie są uruchamiane żadne dalsze komponenty, niezależnie od wartości zwracanej w polu *Kontynuacja*.
- Jeśli parametr *CompCode* nie ma wartości MQCC_OK, zostanie uruchomiony kolejny komponent, chyba że parametr *Continuation* ma wartość MQZCI_STOP.

Obsługiwane są następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_Q_NAME
(2288, X'8F0') Nie znaleziono nazwy kolejki.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-Zakończenie usługi nazw

Ta funkcja jest udostępniana przez komponent usługi nazw i jest uruchamiana przez menedżer kolejek, gdy nie wymaga już usług tego komponentu. Funkcja musi wykonać wszystkie procedury czyszczące wymagane przez komponent.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_TERM_NAME.

Składnia

MQZ_TERM_NAME(*Hconfig* , *Options* , *QMgrName* , *ComponentData* , *CompCode* , *Reason*)

Parametry

Hconfig
Typ: MQHCONFIG-input

Uchwyt konfiguracji. Ten uchwyt reprezentuje konkretny komponent, który jest przerywany. Jest on używany przez komponent podczas wywoływania menedżera kolejek za pomocą funkcji MQZEP.

Opcje
Typ: MQLONG-input

Opcje zakończenia. Musi to być jedna z następujących wartości:

MQZTO_PRIMARY
Zakończenie podstawowe.

MQZTO_SECONDARY
Zakończenie drugorzędne.

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

ComponentData
Typ: MQBYTE x ComponentData, długość-wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżera kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w nim przez dowolną funkcję (w tym funkcję inicjowania) udostępnioną przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Dane komponentu znajdują się w pamięci współużytkowanej dostępnej dla wszystkich procesów.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_NAME.

Po zakończeniu wywołania MQZ_TERM_NAME menedżer kolejek usuwa te dane.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

Niepowodzenie MQRC_TERMINATION_FAILED
(2287, X'8FF') Zakończenie nie powiodło się z niezdefiniowanej przyczyny.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
               &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-kontekst aplikacji

Struktura MQZAC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru *ApplicationContext* . Ten parametr określa dane związane z aplikacją wywołującą.

Tabela 1 zawiera podsumowanie pól w strukturze.

Tabela 838. Pola w tabeli MQZAC

Pole	Opis
StrucId	Identyfikator struktury
Wersja	Numer wersji struktury
ProcessId	Identyfikator procesu
ThreadId	Identyfikator wątku
ApplName	Nazwa aplikacji
UserID	Identyfikator użytkownika
IdentyfikatorEffectiveUser	Efektywny identyfikator użytkownika
Środowisko	Środowisko
CallerType	Typ programu wywołującego

Tabela 838. Pola w tabeli MQZAC (kontynuacja)	
Pole	**Opis**
AuthenticationType	Typ uwierzytelniania
BindType	Typ powiązania

Pola

StrucId
Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZAC_ID_STRUKTURY
Identyfikator struktury kontekstu aplikacji.

Dla języka programowania C zdefiniowana jest również stała MQZAC_STRUC_ID_ARRAY; ma ona taką samą wartość jak MQZAC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja
Typ: MQLONG-input

Numer wersji struktury. Wartość jest następująca:

MQZAC_VERSION_1
Struktura kontekstu aplikacji Version-1 . Stała MQZAC_CURRENT_VERSION określa numer wersji bieżącej.

ProcessId
Typ: MQPID-wejście

Identyfikator procesu aplikacji.

ThreadId
Typ: MQTID-wejście

Identyfikator wątku aplikacji.

ApplName
Typ: MQCHAR28 -dane wejściowe

Nazwa aplikacji.

UserID
Typ: MQCHAR12 -dane wejściowe

Identyfikator użytkownika. W systemie AIX and Linux to pole określa rzeczywisty identyfikator użytkownika aplikacji. W systemie Windows to pole określa identyfikator użytkownika aplikacji.

Identyfikator EffectiveUser
Typ: MQCHAR12 -dane wejściowe

Efektywny identyfikator użytkownika. W systemie AIX and Linux to pole określa efektywny identyfikator użytkownika aplikacji. W systemie Windows to pole jest puste.

Środowisko
Typ: MQLONG-input

Środowisko. To pole określa środowisko, z którego wykonano wywołanie. Pole może przyjmować jedną z następujących wartości:

SERWER MQXE_COMMAND_SERVER
Serwer komend

MQXE_MQSC
Interpreter komendy **runmqsc**

Agent MQXE_MCA
Agent kanału komunikatów MQXE_OTHER

MQXE_INNY
Niezdefiniowane środowisko

CallerType
Typ: MQLONG-input

Typ programu wywołującego. To pole określa typ programu, który wywołał wywołanie. Pole może przyjmować jedną z następujących wartości:

MQXACT_EXTERNAL
Wywołanie jest zewnętrzne względem menedżera kolejek.

MQXACT_INTERNAL
Wywołanie jest wewnętrzne dla menedżera kolejek.

AuthenticationType
Typ: MQLONG-input

Typ uwierzytelniania. To pole określa typ wykonywanego uwierzytelniania. Pole może przyjmować jedną z następujących wartości:

MQZAT_INITIAL_CONTEXT,
Wywołanie uwierzytelniania jest spowodowane zainicjowaniem kontekstu użytkownika. Ta wartość jest używana podczas wywołania MQCONN lub MQCONNX.

MQZAT_CHANGE_CONTEXT,
Wywołanie uwierzytelniania jest spowodowane zmianą kontekstu użytkownika. Ta wartość jest używana, gdy agent MCA zmienia kontekst użytkownika. Temat nadrzędny: MQZAC-

BindType
Typ: MQLONG-input

Typ powiązania. To pole określa typ używanego powiązania. Pole może przyjmować jedną z następujących wartości:

MQCNO_FASTPATH_BINDING,
Powiązanie krótkiej ścieżki.

MQCNO_SHARED_BINDING
Powiązanie współużytkowane.

MQCNO_IZOLOWANE_POWIĄZANIE
Izolowane powiązanie.

Deklaracja C
Zadeklaruj pola struktury w następujący sposób:

```
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-dane uprawnień

Struktura MQZAD jest używana w wywołaniu MQZ_ENUMERATE_AUTHORITY_DATA dla dwóch parametrów: jednego wejścia i jednego wyjścia.

Więcej informacji na temat parametrów **Filter** i **AuthorityBuffer** zawiera sekcja “MQZ_ENUMERATE_AUTHORITY_DATA-wyliczone dane uprawnień” na stronie 1688 :

- Komenda MQZAD jest używana dla parametru **Filter** , który jest przekazywany do wywołania. Ten parametr określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez wywołanie.
- Komenda MQZAD jest również używana dla parametru **AuthorityBuffer** , który jest wyprowadzany z wywołania. Ten parametr określa autoryzacje dla jednej kombinacji nazwy profilu, typu obiektu i jednostki.

Tabela 1. podsumowuje pola w strukturze.

Tabela 839. Pola w MQZAD

Pole	Opis
StrucId	Identyfikator struktury
Wersja	Numer wersji struktury
ProfileName	Nazwa profilu
ObjectType	Typ obiektu
Uprawnienie	Uprawnienie
EntityDataPtr	Wskaźnik do danych jednostki
EntityType	Typ jednostki
Opcje	Opcje

Pola

StrucId
Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

ID_STRUKTURY_MQZAD_STRUCT
Identyfikator struktury danych uprawnień.

Dla języka programowania C zdefiniowana jest również stała MQZAD_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQZAD_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Wersja
Typ: MQLONG-input

Numer wersji struktury. Wartość jest następująca:

MQZAD_VERSION_1
Struktura kontekstu aplikacji Version-1 . Stała MQZAD_CURRENT_VERSION określa numer wersji bieżącej.

Następująca stała określa numer wersji bieżącej:

MQZAD_CURRENT_VERSION
Bieżąca wersja struktury danych uprawnień.

ProfileName
Typ: MQCHAR48 -dane wejściowe

Nazwa profilu.

W przypadku parametru **Filter** pole to określa nazwę profilu, dla którego wymagane są dane uprawnień. Jeśli nazwa jest całkowicie pusta aż do końca pola lub pierwszego znaku o kodzie zero, zwracane są dane uprawnień dla wszystkich nazw profili.

Dla parametru **AuthorityBuffer** to pole jest nazwą profilu, który jest zgodny z podanymi kryteriami wyboru.

ObjectType
Typ: MQLONG-input

Typ obiektu.

Dla parametru **Filter** to pole określa typ obiektu, dla którego wymagane są dane uprawnień. Jeśli wartością jest MQOT_ALL, zwracane są dane uprawnień dla wszystkich typów obiektów.

W przypadku parametru **AuthorityBuffer** pole to określa typ obiektu, do którego ma zastosowanie profil identyfikowany przez parametr **ProfileName** .

Wartość jest jedną z następujących wartości; dla parametru **Filter** poprawna jest również wartość MQOT_ALL:

MQOT_AUTH_INFO
Informacje uwierzytelniające

MQOT_CHANNEL (kanał MQT)
Kanał

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta

MQOT_LISTENER
Program nasłuchujący

LISTA NAZW MQOT_NAMELIST
Lista nazw

PROCES_MQOT
Definicja procesu

MQOT_Q
Kolejka

MQOT_Q_MGR
Menedżer kolejek

USŁUACJA_MQOT
Usługa

Uprawnienie
Typ: MQLONG-input

Uprawnienia.

Dla parametru **Filter** to pole jest ignorowane.

Dla parametru **AuthorityBuffer** to pole reprezentuje autoryzacje, które jednostka ma do obiektów identyfikowanych przez **ProfileName** i **ObjectType**. Jeśli jednostka ma tylko jedno uprawnienie, pole jest równe odpowiedniej wartości autoryzacji (MQZAO_ * stała). Jeśli jednostka ma więcej niż jedno uprawnienie, pole jest bitową wartością OR odpowiednich stałych MQZAO_ *.

EntityDataPtr
Typ: PMQZED-wejście

Adres struktury MQZED identyfikującej jednostkę.

W przypadku parametru **Filter** to pole wskazuje strukturę MQZED identyfikującą jednostkę, dla której wymagane są dane uprawnień. Jeśli **EntityDataPtr** jest wskaźnikiem pustym, zwracane są dane uprawnień dla wszystkich jednostek.

W przypadku parametru **AuthorityBuffer** to pole wskazuje strukturę MQZED identyfikującą jednostkę, dla której zostały zwrócone dane uprawnień.

EntityType
Typ: MQLONG-input

Typ jednostki.

W przypadku parametru **Filter** pole to określa typ jednostki, dla której wymagane są dane uprawnień. Jeśli wartością jest MQZAET_NONE, zwracane są dane uprawnień dla wszystkich typów jednostek.

W przypadku parametru **AuthorityBuffer** to pole określa typ jednostki identyfikowanej przez strukturę MQZED wskazaną przez parametr **EntityDataPtr** .

Jest to jedna z następujących wartości; w przypadku parametru **Filter** poprawna jest także wartość MQZAET_NONE:

MQZAET_PRINCIPAL
Kolumnowo-wierszowa

GRUPA_MQZAET
Grupa

Opcje
Typ: MQAUTHOPT-wejście

Opcje. To pole określa opcje, które dają kontrolę nad wyświetlanymi profilami. Należy podać jedną z następujących wartości:

MQAUTHOPT_NAME_ALL_MATCHING
Wyświetla wszystkie profile

MQAUTHOPT_NAME_EXPLICIT
Wyświetla profile, które mają dokładnie taką samą nazwę, jak określona w polu **ProfileName** .

Ponadto należy również określić jedną z następujących opcji:

MQAUTHOPT_ENTITY_SET
Wyświetl wszystkie profile używane do obliczenia skumulowanego uprawnienia jednostki do obiektu określonego przez parametr **ProfileName** . Parametr **ProfileName** nie może zawierać żadnych znaków wieloznacznych.

- Jeśli określona jednostka jest nazwą użytkownika, dla każdego elementu zestawu {entity, groups} wyświetlany jest najbardziej odpowiedni profil, który ma zastosowanie do obiektu.
- Jeśli określona jednostka jest grupą, wyświetlany jest najbardziej odpowiedni profil z grupy, który ma zastosowanie do obiektu.
- Jeśli ta wartość jest określona, wszystkie wartości parametrów **ProfileName**, **ObjectType**, **EntityType**i nazwa jednostki określone w strukturze **EntityDataPtr** MQZED muszą być niepuste.

Jeśli określono opcję MQAUTHOPT_NAME_ALL_MATCHING, można również określić następującą wartość:

MQAUTHOPT_ENTITY_EXPLICIT
Wyświetla profile, które mają dokładnie taką samą nazwę jednostki jak nazwa jednostki określona w strukturze **EntityDataPtr** MQZED.

Deklaracja C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
```

```
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
                               entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

MQZED-deskryptor jednostki

Struktura MQZED jest używana w wielu wywołaniach usługi autoryzacji w celu określenia jednostki, dla której ma zostać sprawdzona autoryzacja.

Tabela 1. podsumowuje pola w strukturze.

Tabela 840. Pola w MQZED

Pole	Opis
StrucId	Identyfikator struktury
Wersja	Wersja
EntityName Ptr	Nazwa jednostki
EntityDomainPtr	Wskaźnik domeny jednostki
SecurityId	Identyfikator bezpieczeństwa
CorrelationPtr	Wskaźnik korelacji

Pola

StrucId
Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZED_STRUC_ID (identyfikator struktury MQ)
Identyfikator struktury deskryptora jednostki.

W języku programowania C zdefiniowana jest również stała MQZED_STRUC_ID_ARRAY, która ma taką samą wartość jak MQZED_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Wersja
Typ: MQLONG-input

Numer wersji struktury. Wartość jest następująca:

MQZED_VERSION_1
Version-1 struktura deskryptora jednostki.

Następująca stała określa numer wersji bieżącej:

MQZED_CURRENT_VERSION
Bieżąca wersja struktury deskryptora jednostki.

EntityNamePtr
Typ: PMQCHAR-input

Nazwa profilu.

Adres nazwy jednostki. Jest to wskaźnik do nazwy jednostki, której autoryzacja ma zostać sprawdzona.

EntityDomainPtr
Typ: PMQCHAR-input

Adres nazwy domeny jednostki. Jest to wskaźnik do nazwy domeny zawierającej definicję jednostki, której autoryzacja ma zostać sprawdzona.

SecurityId
Typ: MQBYTE40 -dane wejściowe

Uprawnienia.

Identyfikator bezpieczeństwa. Jest to identyfikator bezpieczeństwa, którego autoryzacja ma zostać sprawdzona.

CorrelationPtr
Typ: MQPTR-wejście

Wskaźnik korelacji. Ułatwia to przekazywanie danych korelacji między funkcją uwierzytelniania użytkownika i innymi odpowiednimi funkcjami OAM.

Deklaracja C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

MQZEP-dodawanie punktu wejścia komponentu

Komponent usługi uruchamia tę funkcję podczas inicjowania w celu dodania punktu wejścia do wektora punktu wejścia dla tego komponentu usługi.

Składnia
MQZEP (*Hconfig* , *Funkcja* , *EntryPoint* , *CompCode* , *Przyczyna*)

Parametry

Hconfig
Typ: MQHCONFIG-input

Uchwyt konfiguracji. Ten uchwyt reprezentuje komponent, który jest konfigurowany dla tej konkretnej instalowalnej usługi. Musi być taka sama, jak komponent przekazany do funkcji konfiguracji komponentu przez menedżer kolejek w wywołaniu inicjowania komponentu.

Funkcja
Typ: MQLONG-input

Identyfikator funkcji. Poprawne wartości są definiowane dla każdej instalowalnej usługi.

Jeśli komenda MQZEP jest wywoływana więcej niż raz dla tej samej funkcji, ostatnie wywołanie udostępnia punkt wejścia, który jest używany.

EntryPoint
Typ: PMQFUNC-wejście

Punkt wejścia funkcji. Jest to adres punktu wejścia udostępnionego przez komponent w celu wykonania funkcji.

Wartość NULL jest poprawna i wskazuje, że funkcja nie jest udostępniana przez ten komponent. Dla punktów wejścia, które nie zostały zdefiniowane za pomocą MQZEP, przyjmowana jest wartość NULL.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD FUNKCJI MQRC_FUNCTION_
(2281, X'8E9') Niepoprawny identyfikator funkcji.

BŁĄD MQRC_HCONFIG_ERROR
(2280, X'8E8') Uchwyt konfiguracji jest niepoprawny.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

MQZFP-parametry wolne

Struktura MQZFP jest używana w wywołaniu MQZ_FREE_USER dla parametru *FreeParms* . Ten parametr określa dane związane z zasobem, który ma zostać zwolniony.

Tabela 1. podsumowuje pola w strukturze.

Tabela 841. Pola w MQZFP

Pole	Opis
StrucId	Identyfikator struktury
Wersja	Wersja
Zarezerwowane	Pole zastrzeżone
CorrelationPtr	Wskaźnik korelacji

Pola

StrucId
Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZIC_STRUC_ID (Identyfikator struktury MQ)
Identyfikator struktury kontekstu tożsamości. Dla języka programowania C zdefiniowana jest również stała MQZIC_STRUC_ID_ARRAY, która ma taką samą wartość jak MQZIC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja
Typ: MQLONG-input

Numer wersji struktury. Wartość jest następująca:

MQZFP_VERSION_1
Version-1 wolna struktura parametrów.

Następująca stała określa numer wersji bieżącej:

MQZFP_CURRENT_VERSION
Bieżąca wersja struktury wolnych parametrów.

Zarezerwowane
Typ: MQBYTE8 -dane wejściowe

Pole zastrzeżone. Wartość początkowa jest pusta.

CorrelationPtr
Typ: MQPTR-wejście

Wskaźnik korelacji. Adres danych korelacji związanych z zasobem, który ma zostać zwolniony.

Deklaracja C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

MQZIC-kontekst tożsamości

Struktura MQZIC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru *IdentityContext* .

Struktura MQZIC zawiera informacje o kontekście tożsamości, które identyfikują użytkownika aplikacji, która najpierw umieściła komunikat w kolejce:

- Menedżer kolejek wypełnia pole *UserIdentifier* nazwą identyfikującą użytkownika. Sposób, w jaki menedżer kolejek może to zrobić, zależy od środowiska, w którym działa aplikacja.
- Menedżer kolejek wypełnia pole *AccountingToken* znacznikiem lub numerem określonym przez aplikację, która wstawiła komunikat.
- Aplikacje mogą używać pola *DaneApplIdentity* w celu uzyskania dodatkowych informacji o użytkowniku (na przykład zaszyfrowanego hasła).

Odpowiednio autoryzowane aplikacje mogą ustawić kontekst tożsamości za pomocą funkcji MQZ_AUTHENTICATE_USER.

Identyfikator bezpieczeństwa systemu Windows (SID) jest przechowywany w polu *AccountingToken* podczas tworzenia komunikatu w systemie IBM MQ for Windows. Identyfikator SID może być używany jako uzupełnienie pola *UserIdentifier* i do ustanawiania referencji użytkownika.

Tabela 1. podsumowuje pola w strukturze.

Tabela 842. Pola w MQZIC

Pole	Opis
StrucId	Identyfikator struktury

Tabela 842. Pola w MQZIC (kontynuacja)	
Pole	**Opis**
Wersja	Wersja
UserIdentifier	Identyfikator użytkownika
AccountingToken	Token rozliczania
Dane_tożsamości_aplikacji	Dane tożsamości aplikacji

Pola

StrucId
Typ: MQCHAR4 -dane wejściowe

Identyfikator struktury. Wartość jest następująca:

MQZIC_STRUC_ID (Identyfikator struktury MQ)
Identyfikator struktury kontekstu tożsamości. Dla języka programowania C zdefiniowana jest również stała MQZIC_STRUC_ID_ARRAY, która ma taką samą wartość jak MQZIC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Wersja
Typ: MQLONG-input

Numer wersji struktury. Wartość jest następująca:

MQZIC_VERSION_1
Struktura kontekstu tożsamości Version-1 .

Następująca stała określa numer wersji bieżącej:

MQZIC_CURRENT_VERSION
Bieżąca wersja struktury kontekstu tożsamości.

UserIdentifier
Typ: MQCHAR12 -dane wejściowe

Identyfikator użytkownika. Jest to część kontekstu tożsamości komunikatu. *UserIdentifier* określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu. Więcej informacji na temat pola *UserIdentifier* zawiera sekcja "UserIdentifier (MQCHAR12) dla deskryptora MQMD" na stronie 470.

AccountingToken
Typ: MQBYTE32 -dane wejściowe

Token rozliczania. Jest to część kontekstu tożsamości komunikatu. *AccountingToken* umożliwia aplikacji spowodowanie odpowiedniego obciążenia pracą wykonaną w wyniku wysłania komunikatu. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza ich treści. Więcej informacji na temat pola *AccountingToken* zawiera sekcja "AccountingToken (MQBYTE32) dla MQMD" na stronie 472.

Dane_tożsamości_aplikacji
Typ: MQCHAR32 -dane wejściowe

Dane aplikacji odnoszące się do tożsamości. Jest to część kontekstu tożsamości komunikatu. ApplIdentityDane są informacjami zdefiniowanymi przez pakiet aplikacji, które mogą być używane do udostępniania dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiana przez aplikacje działające z odpowiednimi uprawnieniami użytkownika w celu wskazania, czy dane tożsamości są zaufane. Więcej informacji na temat pola danych ApplIdentityzawiera sekcja "Dane ApplIdentity(MQCHAR32) dla MQMD" na stronie 473.

Deklaracja C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQCHAR12  UserIdentifier;   /* User identifier */
  MQBYTE32  AccountingToken;  /* Accounting token */
  MQCHAR32  ApplIdentityData; /* Application data relating to identity */
};
```

IBM i Informacje uzupełniające o interfejsie instalowalnych usług w systemie IBM i

Te informacje umożliwiają zrozumienie informacji uzupełniających dotyczących usług instalowalnych w systemie IBM i.

Dla każdej funkcji istnieje opis, w tym identyfikator funkcji (dla MQZEP).

Parametry są wyświetlane w kolejności, w jakiej muszą wystąpić. Wszystkie one muszą być obecne.

Po każdej nazwie parametru występuje jego typ danych w nawiasach. Są to podstawowe typy danych opisane w sekcji “Podstawowe typy danych” na stronie 1025.

Po opisie parametrów podano również wywołanie w języku C.

Pojęcia pokrewne

IBM i Instalowalne usługi i komponenty systemu IBM i

ALW Instalowalne usługi i komponenty dla systemów UNIX, Linux i Windows

Odsyłacze pokrewne

“Informacje uzupełniające o interfejsie usług instalowalnych” na stronie 1671
Ta kolekcja tematów zawiera informacje uzupełniające na temat usług instalowalnych.

IBM i MQZEP (dodawanie punktu wejścia komponentu) w systemie IBM i

Ta funkcja jest wywoływana przez komponent usługi podczas inicjowania w celu dodania punktu wejścia do wektora punktu wejścia dla tego komponentu usługi.

Składnia

```
MQZEP (Hconfig, Function, EntryPoint, CompCode, Reason)
```

Parametry

Wywołanie MQZEP ma następujące parametry.

Hconfig (MQHCONFIG)-wejście
Uchwyt konfiguracji.

Ten uchwyt reprezentuje komponent, który jest konfigurowany dla tej konkretnej instalowalnej usługi. Musi być taka sama, jak ta, która została przekazana do funkcji konfiguracji komponentu przez menedżer kolejek w wywołaniu inicjowania komponentu.

Funkcja (MQLONG)-wejście
Identyfikator funkcji.

Poprawne wartości są definiowane dla każdej instalowalnej usługi. Jeśli funkcja MQZEP jest wywoływana więcej niż raz dla tej samej funkcji, ostatnie wywołanie udostępnia punkt wejścia, który jest używany.

EntryPoint (PMQFUNC)-wejście
Punkt wejścia funkcji.

Jest to adres punktu wejścia udostępnionego przez komponent w celu wykonania funkcji. Wartość NULL jest poprawna i wskazuje, że funkcja nie jest udostępniana przez ten komponent. Dla punktów wejścia, które nie zostały zdefiniowane przy użyciu parametru MQZEP, przyjmowana jest wartość NULL .

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD FUNKCJI MQRC_FUNCTION_
(2281, X'8E9') Niepoprawny identyfikator funkcji.

BŁĄD MQRC_HCONFIG_ERROR
(2280, X'8E8') Uchwyt konfiguracji jest niepoprawny.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQHCONFIG  Hconfig;     /* Configuration handle */
MQLONG     Function;    /* Function identifier */
PMQFUNC    EntryPoint;  /* Function entry point */
MQLONG     CompCode;    /* Completion code */
MQLONG     Reason;      /* Reason code qualifying CompCode */
```

IBM i MQHCONFIG (uchwyt konfiguracji) w systemie IBM i

Typ danych MQHCONFIG reprezentuje uchwyt konfiguracji, czyli komponent, który jest konfigurowany dla konkretnej instalowalnej usługi. Uchwyt konfiguracji musi być wyrównany do swojej granicy naturalnej.

Aplikacje muszą testować zmienne tego typu tylko pod kątem równości.

Deklaracja C

```
typedef void MQPOINTER MQHCONFIG;
```

IBM i PMQFUNC (wskaźnik do funkcji) w systemie IBM i

Wskaźnik do funkcji.

Deklaracja C

```
typedef void MQPOINTER PMQFUNC;
```

IBM i MQZ_AUTHENTICATE_USER (uwierzytelnienie użytkownika) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 . Jest on wywoływany przez menedżer kolejek w celu uwierzytelnienia użytkownika lub ustawienia pól kontekstu tożsamości.

Jest on wywoływany po ustanowieniu kontekstu aplikacji użytkownika IBM MQ . Dzieje się tak podczas wywołań połączenia w punkcie, w którym inicjowany jest kontekst użytkownika aplikacji, i w każdym punkcie, w którym zmieniany jest kontekst użytkownika aplikacji. Za każdym razem, gdy wykonywane jest wywołanie połączenia, informacje o kontekście użytkownika aplikacji są ponownie uzyskiwane w polu *IdentityContext* .

Identyfikator tej funkcji (dla MQZEP) to MQZID_AUTHENTICATE_USER.

Składnia

MQZ_AUTHENTICATE_USER ***(QMgrName, SecurityParms, ApplicationContext,***
IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_AUTHENTICATE_USER ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero. Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

SecurityParms (MQCSP)-dane wejściowe
Parametry zabezpieczeń.

Dane dotyczące identyfikatora użytkownika, hasła i typu uwierzytelniania.

Podczas wywołania MQCONN MQI ten parametr zawiera wartość NULL lub wartości domyślne.

ApplicationContext (MQZAC)-wejście
Kontekst aplikacji.

Dane dotyczące aplikacji wywołującej. Szczegółowe informacje można znaleźć w sekcji "MQZAC (kontekst aplikacji) w systemie IBM i" na stronie 1767. Podczas każdego wywołania MQCONN lub MQCONNX MQI ponownie uzyskiwane są informacje o kontekście użytkownika w strukturze MQZAC.

IdentityContext (MQZIC)-wejście/wyjście
Kontekst tożsamości.

Na wejściu do funkcji uwierzytelniania użytkownika identyfikuje ona bieżący kontekst tożsamości. Funkcja uwierzytelniania użytkownika może to zmienić. W tym momencie menedżer kolejek przyjmuje nowy kontekst tożsamości. Więcej informacji na temat struktury MQZIC zawiera sekcja "MQZIC (kontekst tożsamości) w systemie IBM i" na stronie 1773 .

CorrelationPtr (MQPTR)-dane wyjściowe
Wskaźnik korelacji.

Określa adres wszystkich danych korelacji. Wskaźnik ten jest następnie przekazywany do innych wywołań OAM.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane podczas następnego wywołania funkcji tego komponentu. Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Flaga kontynuacji.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależna od innych składników.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                   IdentityContext, &CorrelationPtr, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQCSP     SecurityParms;      /* Security parameters */
MQZAC     ApplicationContext; /* Application context */
MQZIC     IdentityContext;    /* Identity context */
MQPTR     CorrelationPtr;     /* Correlation pointer */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

IBM i MQZ_CHECK_AUTHORITY (uprawnienie do sprawdzania) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu sprawdzenia, czy jednostka ma uprawnienia do wykonywania określonych działań na określonym obiekcie.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_CHECK_AUTHORITY.

Składnia

MQZ_CHECK_AUTHORITY *(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

Parametry

Wywołanie MQZ_CHECK_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero. Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe
Nazwa jednostki.

Nazwa jednostki, której uprawnienia do obiektu mają zostać sprawdzone. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

Nie jest konieczne, aby ta jednostka była znana bazowej usłudze zabezpieczeń. Jeśli nie jest znana, do sprawdzenia używane są autoryzacje specjalnej grupy **nobody** (nikt), do której należą wszystkie obiekty. Pusta nazwa jest poprawna i może być użyta w ten sposób.

EntityType (MQLONG)-dane wejściowe
Typ jednostki.

Typ jednostki określony przez *EntityName*. Jest to jedna z poniższych nazw:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName (MQCHAR48)-dane wejściowe
Nazwa obiektu.

Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType (MQLONG)-dane wejściowe
Typ obiektu.

Typ jednostki określony przez *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

Uprawnienie (MQLONG)-dane wejściowe
Uprawnienie do sprawdzenia.

Jeśli sprawdzana jest jedna autoryzacja, to pole jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli sprawdzana jest więcej niż jedna autoryzacja, jest to bitowe LUB odpowiednich stałych MQZAO_ *.

Do korzystania z wywołań MQI mają zastosowanie następujące autoryzacje:

MQZAO_CONNECT
Możliwość użycia wywołania MQCONN.

MQZAO_BROWSE,
Możliwość użycia wywołania MQGET z opcją przeglądania.

Umożliwia to określenie opcji MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR lub MQGMO_BROWSE_NEXT w wywołaniu MQGET.

MQZAO_INPUT
Możliwość użycia wywołania MQGET z opcją wejściową.

Umożliwia to określenie opcji MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE lub MQOO_INPUT_AS_Q_DEF w wywołaniu MQOPEN.

MQZAO_WYNIK
Możliwość użycia wywołania MQPUT.

Umożliwia to określenie opcji MQOO_OUTPUT w wywołaniu MQOPEN.

MQZAO_INQUIRE,
Możliwość użycia wywołania MQINQ.

Umożliwia to określenie opcji MQOO_INQUIRE w wywołaniu MQOPEN.

MQZAO_SET
Możliwość użycia wywołania MQSET.

Umożliwia to określenie opcji MQOO_SET w wywołaniu MQOPEN.

MQZAO_PASS_IDENTITY_CONTEXT,
Możliwość przekazywania kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_PASS_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_IDENTITY_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_PASS_ALL_CONTEXT (mqzao_pass_all)
Możliwość przekazania całego kontekstu.

Umożliwia to określenie opcji MQOO_PASS_ALL_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_PASS_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_IDENTITY_CONTEXT,
Możliwość ustawienia kontekstu tożsamości.

Umożliwia to określenie opcji MQOO_SET_IDENTITY_CONTEXT w wywołaniu MQOPEN oraz opcji MQPMO_SET_IDENTITY_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_SET_ALL_CONTEXT (mqzao_set_all)
Możliwość ustawienia całego kontekstu.

Umożliwia to określenie opcji MQOO_SET_ALL_CONTEXT w wywołaniu MQOPEN i opcji MQPMO_SET_ALL_CONTEXT w wywołaniach MQPUT i MQPUT1 .

MQZAO_ALTERNATE_USER_AUTHORITY (uprawnienie użytkownika na przemian)
Możliwość korzystania z alternatywnych uprawnień użytkownika.

Umożliwia to określenie opcji MQOO_ALTERNATE_USER_AUTHORITY w wywołaniu MQOPEN, a opcji MQPMO_ALTERNATE_USER_AUTHORITY w wywołaniu MQPUT1 .

MQZAO_ALL_MQI
Wszystkie autoryzacje MQI.

Spowoduje to włączenie wszystkich autoryzacji opisanych wcześniej.

Do administrowania menedżerem kolejek mają zastosowanie następujące autoryzacje:

MQZAO_CREATE
Możliwość tworzenia obiektów określonego typu.

MQZAO_USUŃ
Możliwość usunięcia określonego obiektu.

MQZAO_DISPLAY
Możliwość wyświetlania atrybutów określonego obiektu.

MQZAO_CHANGE
Możliwość zmiany atrybutów określonego obiektu.

MQZAO_CLEAR,
Możliwość usuwania wszystkich komunikatów z określonej kolejki.

Autoryzacja MQZAO_AUTORYZACJI
Możliwość autoryzowania innych użytkowników dla określonego obiektu.

MQZAO_CONTROL
Możliwość uruchamiania, zatrzymywania lub wysyłania pakietów ping do obiektu kanału innego niż kanał klienta.

MQZAO_CONTROL_EXTENDED,
Możliwość zresetowania numeru kolejnego lub rozstrzygnięcia wątpliwego komunikatu w obiekcie kanału innego niż kanał klienta.

MQZAO_ALL_ADMIN
Wszystkie autoryzacje administracyjne inne niż MQZAO_CREATE.

Następujące autoryzacje mają zastosowanie zarówno do używania interfejsu MQI, jak i do administrowania menedżerem kolejek:

MQZAO_WSZYSTKIE
Wszystkie autoryzacje inne niż MQZAO_CREATE.

MQZAO_BRAK
Brak autoryzacji.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak dla MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
```

```
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_CHECK_PRIVILEGED-Sprawdź, czy użytkownik jest uprzywilejowany

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_6 i jest wywoływana przez menedżer kolejek w celu określenia, czy określony użytkownik jest użytkownikiem uprzywilejowanym.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_CHECK_PRIVILEGED.

Składnia

MQZ_CHECK_PRIVILEGED(*QMgrName* , *EntityData* , *EntityType* , *ComponentData* , *Continuation* , *CompCode* , *Reason*)

Parametry

QMgrName
Typ: MQCHAR48 -dane wejściowe

Nazwa menedżera kolejek. Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

EntityData
Typ: MQZED-wejście

Dane jednostki. Dane dotyczące jednostki, która ma zostać sprawdzona. Więcej informacji na ten temat zawiera sekcja "MQZED-deskryptor jednostki" na stronie 1730.

EntityType
Typ: MQLONG-input

Typ jednostki. Typ jednostki określony przez EntityData. Musi to być jedna z następujących wartości:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ComponentData
Typ: MQBYTExComponentDataLength -wejście/wyjście

Dane komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z tych funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja
Typ: MQLONG-wyjście

Wskaźnik kontynuacji ustawiony przez komponent. Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_CHECK_AUTHORITY ma to taki sam efekt, jak w przypadku MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

Jeśli wywołanie komponentu nie powiedzie się (*CompCode* zwraca wartość MQCC_FAILED), a parametr *Continuation* ma wartość MQZCI_DEFAULT lub MQZCI_CONTINUE, menedżer kolejek kontynuuje wywoływanie innych komponentów (jeśli istnieją).

Jeśli wywołanie powiedzie się (czyli kod *CompCode* zwróci wartość MQCC_OK), żadne inne komponenty nie zostaną wywołane bez względu na ustawienie opcji *Kontynuacja* .

Jeśli wywołanie nie powiedzie się, a parametr *Continuation* ma wartość MQZCI_STOP, nie zostaną wywołane żadne inne komponenty, a błąd zostanie zwrócony do menedżera kolejek. Komponenty nie mają informacji o poprzednich wywołaniach, dlatego parametr *Kontynuacja* jest zawsze ustawiany na wartość MQZCI_DEFAULT przed wywołaniem.

CompCode
Typ: MQLONG-wyjście

Kod zakończenia. Musi to być jedna z następujących wartości:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna
Typ: MQLONG-wyjście

Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

MQRC_NOT_PRIVILEGED
(2584, X'A18') Ten użytkownik nie jest użytkownikiem uprzywilejowanym.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Kody zakończenia i kody przyczyny interfejsu API.

Wywołanie C

```
MQZ_CHECK_PRIVILEGED (QMgrName, &EntityData, EntityType,
                      ComponentData, &Continuation,
                      &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQZED     EntityData;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQBYTE    ComponentData[n];  /* Component data */
```

```
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_COPY_ALL_AUTHORITY (Kopiowanie wszystkich uprawnień) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji. Jest ona wywoływana przez menedżer kolejek w celu skopiowania wszystkich autoryzacji, które są obecnie obowiązujące dla obiektu odniesienia, do innego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_COPY_ALL_AUTHORITY.

Składnia

MQZ_COPY_ALL_AUTHORITY ***(QMgrName, RefObjectName, ObjectName,***
ObjectType, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_COPY_ALL_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

RefObjectNazwa (MQCHAR48)-wejście
Nazwa obiektu odniesienia.

Nazwa obiektu odniesienia, którego autoryzacje mają zostać skopiowane. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

ObjectName (MQCHAR48)-dane wejściowe
Nazwa obiektu.

Nazwa obiektu, dla którego mają zostać ustawione dostępy. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

ObjectType (MQLONG)-dane wejściowe
Typ obiektu.

Typ obiektu określony przez *RefObjectName* i *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla MQZ_COPY_ALL_AUTHORITY ma to taki sam efekt, jak dla MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_REF_OBJECT
(2294, X'8F6') Nieznany obiekt odniesienia.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  RefObjectName;     /* Reference object name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_DELETE_AUTHORITY (uprawnienie do usuwania) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest wywoływana przez menedżer kolejek w celu usunięcia wszystkich autoryzacji powiązanych z określonym obiektem.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_DELETE_AUTHORITY.

Składnia

MQZ_DELETE_AUTHORITY ***(QMgrName, ObjectName, ObjectType,***
ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_DELETE_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

ObjectName (MQCHAR48)-dane wejściowe
Nazwa obiektu.

Nazwa obiektu, dla którego mają zostać usunięte dostępy. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType (MQLONG)-dane wejściowe
Typ obiektu.

Typ jednostki określony przez *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla MQZ_DELETE_AUTHORITY ma to taki sam efekt jak MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_ENUMERATE_AUTHORITY_DATA (Enumerate authority data) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_4 i jest wielokrotnie wywoływana przez menedżer kolejek w celu pobrania wszystkich danych uprawnień zgodnych z kryteriami wyboru określonymi w pierwszym wywołaniu.

Identyfikator tej funkcji (dla MQZEP) to MQZID_ENUMERATE_AUTHORITY_DATA.

Składnia

MQZ_ENUMERATE_AUTHORITY_DATA ***(QMgrName, StartEnumeration,***
Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength,
ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_ENUMERATE_AUTHORITY_DATA ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

StartEnumeration (MQLONG)-wejście
Flaga wskazująca, czy wywołanie powinno rozpocząć wyliczanie.

Wskazuje, czy wywołanie powinno rozpocząć wyliczanie danych uprawnień, czy też kontynuować wyliczanie danych uprawnień rozpoczęte przez poprzednie wywołanie MQZ_ENUMERATE_AUTHORITY_DATA. Jest to jedna z następujących wartości:

MQZSE_START
Rozpocznij wyliczanie.

Wywołanie jest wywoływane z tą wartością w celu rozpoczęcia wyliczania danych uprawnień. Parametr **Filter** określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez to i kolejne wywołania.

MQZSE_CONTINUE
Kontynuuj wyliczanie.

Wywołanie jest wywoływane z tą wartością, aby kontynuować wyliczanie danych uprawnień. W tym przypadku parametr **Filter** jest ignorowany i może zostać określony jako wskaźnik pusty

(kryteria wyboru są określone przez parametr **Filter** określony przez wywołanie, które miało parametr *StartEnumeration* ustawiony na wartość MQZSE_START).

Filtr (MQZAD)-wejście
Filtr.

Jeśli parametr *StartEnumeration* ma wartość MQZSE_START, parametr *Filter* określa kryteria wyboru, które mają zostać użyte do wybrania zwracanych danych uprawnień. Jeśli *Filter* jest wskaźnikiem pustym, nie są używane żadne kryteria wyboru, tzn. zwracane są wszystkie dane uprawnień. Szczegółowe informacje na temat kryteriów wyboru, które mogą być używane, zawiera sekcja "MQZAD (dane uprawnień) w systemie IBM i" na stronie 1769 .

Jeśli parametr *StartEnumeration* ma wartość MQZSE_CONTINUE, parametr *Filter* jest ignorowany i można go określić jako wskaźnik pusty.

AuthorityBufferDługość (MQLONG)-wejście
Długość *AuthorityBuffer*.

Jest to długość parametru **AuthorityBuffer** w bajtach. Bufor uprawnień musi być wystarczająco duży, aby pomieścić zwracane dane.

AuthorityBuffer (MQZAD)-dane wyjściowe
Dane uprawnień.

Jest to bufor, w którym zwracane są dane uprawnień. Bufor musi być wystarczająco duży, aby pomieścić strukturę MQZAD, strukturę MQZED oraz najdłuższą nazwę jednostki i najdłuższą zdefiniowaną nazwę domeny.

Uwaga: Ten parametr jest zdefiniowany jako MQZAD, ponieważ MQZAD zawsze występuje na początku buforu. Jeśli jednak bufor jest zadeklarowany jako MQZAD, będzie on zbyt mały-musi być większy niż MQZAD, aby mógł pomieścić nazwy MQZAD, MQZED oraz nazwy jednostek i domen.

AuthorityDataDługość (MQLONG)-dane wyjściowe
Długość danych zwracanych w *AuthorityBuffer*.

Jest to długość danych zwracanych w produkcie *AuthorityBuffer*. Jeśli bufor uprawnień jest zbyt mały, parametr *AuthorityDataLength* jest ustawiony na wymaganą długość buforu, a wywołanie zwraca kod zakończenia MQCC_FAILED i kod przyczyny MQRC_BUFFER_LENGTH_ERROR.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla MQZ_ENUMERATE_AUTHORITY_DATA ma to taki sam efekt jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR
(2005, X'7D5') Niepoprawny parametr długości buforu.

MQRC_NO_DATA_AVAILABLE
(2379, X'94B') Brak dostępnych danych.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
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-wolny użytkownik

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest wywoływana przez menedżer kolejek w celu zwolnienia powiązanego przydzielonego zasobu. Jest ona wywoływana po zakończeniu działania aplikacji we wszystkich kontekstach użytkownika, na przykład podczas wywołania MQI MQDISC.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_FREE_USER.

IBM i MQZ_GET_AUTHORITY (pobieranie uprawnień) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu pobrania uprawnienia, które jednostka ma, aby uzyskać dostęp do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_GET_AUTHORITY.

Składnia

MQZ_GET_AUTHORITY (***QMgrName, EntityName, EntityType, ObjectName,***
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_GET_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe
Nazwa jednostki.

Nazwa jednostki, której dostęp do obiektu ma zostać pobrany. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

EntityType (MQLONG)-dane wejściowe
Typ jednostki.

Typ jednostki określony przez *EntityName*. Można podać następującą wartość:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName (MQCHAR48)-dane wejściowe
Nazwa obiektu.

Nazwa obiektu, dla którego mają zostać pobrane uprawnienia jednostki. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType (MQLONG)-dane wejściowe
Typ obiektu.

Typ jednostki określony przez *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

Uprawnienie (MQLONG)-dane wyjściowe
Organ podmiotu.

Jeśli jednostka ma jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ma więcej niż jedno uprawnienie, to pole jest bitową OR odpowiednich stałych MQZAO_ *.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku komendy MQZ_GET_AUTHORITY ma to taki sam efekt, jak w przypadku komendy MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_GET_EXPLICIT_AUTHORITY (pobranie jawnego uprawnienia) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu pobrania uprawnienia, które grupa nazwana ma do uzyskania dostępu do określonego obiektu (ale bez dodatkowych uprawnień grupy **nobody**) lub uprawnienia, które grupa podstawowa określonej nazwy użytkownika ma do uzyskania dostępu do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_GET_EXPLICIT_AUTHORITY.

Składnia

MQZ_GET_EXPLICIT_AUTHORITY ***(QMgrName, EntityName, EntityType,***
ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_GET_EXPLICIT_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe
Nazwa jednostki.

Nazwa jednostki, z której ma zostać pobrany dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

EntityType (MQLONG)-dane wejściowe
Typ jednostki.

Typ jednostki określony przez *EntityName*. Można podać następującą wartość:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName (MQCHAR48)-dane wejściowe
Nazwa obiektu.

Nazwa obiektu, dla którego mają zostać pobrane uprawnienia jednostki. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType (MQLONG)-dane wejściowe
Typ obiektu.

Typ jednostki określony przez *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

Uprawnienie (MQLONG)-dane wyjściowe
Organ podmiotu.

Jeśli jednostka ma jedno uprawnienie, to pole jest równe odpowiedniej operacji autoryzacji (MQZAO_ * stała). Jeśli ma więcej niż jedno uprawnienie, to pole jest bitową OR odpowiednich stałych MQZAO_ *.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla opcji MQZ_GET_EXPLICIT_AUTHORITY ma to taki sam efekt, jak opcja MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
```

```
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority of entity */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_INIT_AUTHORITY (Inicjowanie usługi autoryzacji) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest wywoływana przez menedżer kolejek podczas konfigurowania komponentu. Oczekuje się wywołania MQZEP w celu udostępnienia informacji menedżerowi kolejek.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INIT_AUTHORITY.

Składnia

MQZ_INIT_AUTHORITY (***Hconfig, Options, QMgrName, ComponentDataLength,***
ComponentData, Version, CompCode, Reason)

Parametry

Wywołanie MQZ_INIT_AUTHORITY ma następujące parametry.

Hconfig (MQHCONFIG)-wejście
Uchwyt konfiguracji.

Ten uchwyt reprezentuje konkretny inicjowany komponent. Jest on używany przez komponent podczas wywoływania menedżera kolejek za pomocą funkcji MQZEP.

Opcje (MQLONG)-wejście
Opcje inicjowania.

Jest to jedna z poniższych nazw:

Podstawowy MQZIO_PRIMARY
Inicjowanie podstawowe.

MQZIO_SECONDARY
Inicjowanie dodatkowe.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

ComponentData-długość (MQLONG)-wejście
Długość danych komponentu.

Długość w bajtach obszaru *ComponentData* . Ta długość jest zdefiniowana w danych konfiguracyjnych komponentu.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Jest on inicjowany do samych zer przed wywołaniem podstawowej funkcji inicjowania komponentu. Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w nim przez dowolną funkcję (w tym funkcję inicjowania) udostępnioną przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu funkcji tego komponentu.

Wersja (MQLONG)-wejście/wyjście
Numer wersji.

Na wejściu funkcji inicjowania identyfikuje ona *najwyższy* numer wersji obsługiwany przez menedżer kolejek. Funkcja inicjowania musi zmienić tę wersję, jeśli jest to konieczne, na wersję interfejsu obsługiwaną przez *interfejs* . Jeśli po zwróceniu menedżer kolejek nie obsługuje wersji zwróconej przez komponent, wywołuje funkcję MQZ_TERM_AUTHORITY komponentu i nie korzysta już z tego komponentu.

Obsługiwane są następujące wartości:

MQZAS_VERSION_1
Wersja 1.

MQZAS_VERSION_2
Wersja 2.

MQZAS_VERSION_3
Wersja 3.

MQZAS_VERSION_4
Wersja 4.

MQZAS_VERSION_5
Wersja 5.

MQZAS_VERSION_6
IBM WebSphere MQ 6.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

Niepowodzenie MQRC_INITIALIZATION_FAILED
(2286, X'8EE') Inicjowanie nie powiodło się z niezdefiniowanej przyczyny.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;              /* Configuration handle */
MQLONG     Options;              /* Initialization options */
```

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     ComponentDataLength;  /* Length of component data */
MQBYTE     ComponentData[n];     /* Component data */
MQLONG     Version;              /* Version number */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

IBM i MQZ_INQUIRE (zapytanie o usługę autoryzacji) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_5 i jest wywoływana przez menedżer kolejek w celu wykonania zapytania o obsługiwaną funkcję. Jeśli używanych jest wiele komponentów usług, są one wywoływane w kolejności odwrotnej do kolejności, w jakiej zostały zainstalowane.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_INQUIRE.

Składnia

MQZ_INQUIRE
(QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_INQUIRE ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

SelectorCount (MQLONG)-wejście
Liczba selektorów.

Liczba selektorów podanych w parametrze Selektory.

Wartość musi należeć do zakresu od 0 do 256.

Selektory (MQLONG x SelectorCount)-wejście
Selektory.

Tablica selektorów. Każdy selektor identyfikuje wymagany atrybut i musi mieć jeden z następujących typów:

- MQIACF_ * (liczba całkowita)
- MQCACF_ * (znak)

Selektory można określić w dowolnej kolejności. Liczba selektorów w tablicy jest wskazywana przez parametr SelectorCount .

Atrybuty całkowitoliczbowe identyfikowane przez selektory są zwracane w parametrze IntAttrs w tej samej kolejności, w jakiej występują w selektorach.

Atrybuty znakowe identyfikowane przez selektory są zwracane w parametrze CharAttrs w tej samej kolejności, w jakiej występują w selektorach.

IntAttrLiczba (MQLONG)-dane wejściowe
Liczba atrybutów całkowitych.

Liczba atrybutów całkowitych podana w parametrze IntAttrs .

Wartość musi być z zakresu od 0 do 256.

IntAttrs (MQLONG x IntAttrCount)-dane wyjściowe
Atrybuty całkowitoliczbowe.

Tablica atrybutów całkowitych. Atrybuty całkowitoliczbowe są zwracane w tej samej kolejności, co odpowiadające im selektory liczb całkowitych w tablicy Selektory.

CharAttrLiczba (MQLONG)-dane wejściowe
Długość buforu atrybutów znakowych.

Długość parametru CharAttrs w bajtach.

Wartość musi zawierać co najmniej sumę długości żądanych atrybutów znakowych. Jeśli nie są żądane żadne atrybuty znakowe, zero jest poprawną wartością.

CharAttrs (MQLONG x CharAttrliczba)-dane wyjściowe
Bufor atrybutów znakowych.

Bufor zawierający atrybuty znakowe, skonkatenowany ze sobą. Atrybuty znakowe są zwracane w tej samej kolejności, co odpowiadające im selektory znakowe w tablicy Selektory.

Długość buforu jest określona przez parametr liczby CharAttr.

SelectorReturned (MQLONGxSelector)-dane wejściowe
Selektor został zwrócony.

Tablica wartości identyfikująca, które atrybuty zostały zwrócone z zestawu żądanego przez selektory w parametrze Selektory. Liczba wartości w tej tablicy jest wskazywana przez parametr SelectorCount . Każda wartość w tablicy jest powiązana z selektorem z odpowiedniej pozycji w tablicy selektorów. Każda wartość jest jedną z następujących:

MQZSL_XX_ENCODE_CASE_ONE zwrócona
Atrybut żądany przez odpowiedni selektor w parametrze Selektory został zwrócony.

MQZSL_NIE_ZWRÓCONO
Atrybut żądany przez odpowiedni selektor w parametrze Selektory nie został zwrócony.

Tablica jest inicjowana ze wszystkimi wartościami *MQZSL_NOT_RETURNED*. Gdy komponent usługi autoryzacji zwraca atrybut, ustawia odpowiednią wartość w tablicy na *MQZSL_RETURNED*. Umożliwia to innym komponentom usługi autoryzacji, do których jest wykonywane wywołanie zapytania, identyfikowanie, które atrybuty zostały już zwrócone.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Flaga kontynuacji.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależna od innych składników.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

Ostrzeżenie MQCC
Częściowe zakończenie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT
Za mało miejsca na atrybuty znakowe.

MQRC_INT_COUNT_TOO_SMALL
Brak wystarczającej ilości miejsca dla atrybutów całkowitoliczbowych.

Jeśli *CompCode* ma wartość MQCC_FAILED:

BŁĄD MQRC_SELECTOR_COUNT_ERROR
Liczba selektorów jest niepoprawna.

BŁĄD WYWOŁANIA MQRC_SELECTOR_ERROR
Selektor atrybutu jest niepoprawny.

MQRC_SELECTOR_LIMIT_EXCEEDED
Określono zbyt wiele selektorów.

BŁĄD MQRC_INT_ATTR_COUNT_ERROR
Niepoprawna liczba atrybutów całkowitych.

BŁĄD TABELI MQRC_INT_ATTRS_ARRAY_ERROR
Niepoprawna tablica atrybutów całkowitoliczbowych.

BŁĄD MQRC_CHAR_ATTR_LENGTH_ERROR
Liczba atrybutów znakowych jest niepoprawna.

BŁĄD MQRC_CHAR_ATTRS_ERROR
Łańcuch atrybutów znakowych jest niepoprawny.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Wywołanie C

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
                  &IntAttrs, CharAttrLength, &CharAttrs,
                  SelectorReturned, ComponentData, &Continuation,
                  &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     SelectorCount;         /* Selector count */
MQLONG     Selectors[n];          /* Selectors */
MQLONG     IntAttrCount;          /* IntAttrs count */
MQLONG     IntAttrs[n];           /* Integer attributes */
MQLONG     CharAttrCount;         /* CharAttrs count */
MQLONG     CharAttrs[n];          /* Chatacter attributes */
MQLONG     SelectorReturned[n];   /* Selector returned */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Continuation;          /* Continuation indicator set by
                                     component */
```

```
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

IBM i MQZ_REFRESH_CACHE (Odśwież wszystkie autoryzacje) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_3 . Jest on wywoływany przez menedżer kolejek w celu odświeżenia listy autoryzacji przechowywanych wewnętrznie przez komponent.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_REFRESH_CACHE (8L).

Składnia

MQZ_REFRESH_CACHE
(QMgrName, ComponentData, Continuation, CompCode, Reason)

Parametry

***QMgrName* (MQCHAR48)-wejście**
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent używał tej nazwy w zdefiniowany sposób.

***ComponentData* (MQBYTE x *ComponentDatadługość*) -wejście/wyjście**
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Wszelkie zmiany wprowadzone w nim przez funkcje udostępniane przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu funkcji komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze *ComponentDataLength* wywołania MQZ_INIT_AUTHORITY.

***Kontynuacja* (MQLONG)-dane wyjściowe**
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

W przypadku MQZ_REFRESH_CACHE efekt jest taki sam jak w przypadku MQZCI_CONTINUE.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

***CompCode* (MQLONG)-dane wyjściowe**
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

***Przyczyna* (MQLONG)-dane wyjściowe**
Kod przyczyny kwalifikujący kod *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli kod *CompCode* to MQCC_FAILED:

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

Wywołanie C

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Zadeklaruj parametry w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZ_SET_AUTHORITY (ustawianie uprawnień) w systemie IBM i

Ta funkcja jest udostępniana przez komponent usługi autoryzacji MQZAS_VERSION_1 i jest wywoływana przez menedżer kolejek w celu ustawienia uprawnienia dostępu jednostki do określonego obiektu.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_SET_AUTHORITY.

Uwaga: Ta funkcja nadpisuje wszystkie istniejące uprawnienia. Aby zachować istniejące uprawnienia, należy je ponownie ustawić za pomocą tej funkcji.

Składnia

MQZ_SET_AUTHORITY ***(QMgrName, EntityName, EntityType, ObjectName,***
ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)

Parametry

Wywołanie MQZ_SET_AUTHORITY ma następujące parametry.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

EntityName (MQCHAR12)-dane wejściowe
Nazwa jednostki.

Nazwa jednostki, dla której ma zostać ustawiony dostęp do obiektu. Maksymalna długość łańcucha wynosi 12 znaków; jeśli jest krótszy, jest dopełniany do prawej strony spacjami. Nazwa nie jest zakończona znakiem o kodzie zero.

EntityType (MQLONG)-dane wejściowe
Typ jednostki.

Typ jednostki określony przez *EntityName*. Można podać następującą wartość:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

ObjectName (MQCHAR48)-dane wejściowe
Nazwa obiektu.

Nazwa obiektu, do którego wymagany jest dostęp. Maksymalna długość łańcucha wynosi 48 znaków; jeśli jest on krótszy, jest dopełniany do prawej strony odstępami. Nazwa nie jest zakończona znakiem o kodzie zero.

Jeśli *ObjectType* to MQOT_Q_MGR, ta nazwa jest taka sama jak *QMgrName*.

ObjectType (MQLONG)-dane wejściowe
Typ obiektu.

Typ jednostki określony przez *ObjectName*. Jest to jedna z poniższych nazw:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

Uprawnienie (MQLONG)-dane wejściowe
Uprawnienie do sprawdzenia.

Jeśli ustawiona jest jedna autoryzacja, to pole jest równe odpowiedniej operacji autoryzacji (stała MQZAO_ *). Jeśli ustawiona jest więcej niż jedna autoryzacja, jest to bitowe LUB odpowiednich stałych MQZAO_ *.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Kontynuacja (MQLONG)-dane wyjściowe
Wskaźnik kontynuacji ustawiony przez komponent.

Można podać następujące wartości:

MQZCI_DEFAULT
Kontynuacja zależy od menedżera kolejek.

Dla MQZ_SET_AUTHORITY ma to taki sam efekt, jak dla MQZCI_STOP.

MQZCI_CONTINUE
Przejdź do następnego komponentu.

MQZCI_STOP
Nie kontynuuj od następnego komponentu.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_NOT_AUTHORIZED (nieautoryzowany)
(2035, X'7F3') Brak uprawnień dostępu.

BŁĄD MQRC_SERVICE_ERROR
(2289, X'8F1') Wystąpił nieoczekiwany błąd podczas uzyskiwania dostępu do usługi.

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

MQRC_UNKNOWN_ENTITY
(2292, X'8F4') Jednostka nieznana do obsługi.

Wywołanie C

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQCHAR48  QMgrName;          /* Queue manager name */
MQCHAR12  EntityName;        /* Entity name */
MQLONG    EntityType;        /* Entity type */
MQCHAR48  ObjectName;        /* Object name */
MQLONG    ObjectType;        /* Object type */
MQLONG    Authority;         /* Authority to be checked */
MQBYTE    ComponentData[n];  /* Component data */
MQLONG    Continuation;      /* Continuation indicator set by
                                component */
MQLONG    CompCode;          /* Completion code */
MQLONG    Reason;            /* Reason code qualifying CompCode */
```

MQZ_TERM_AUTHORITY-Zakończenie usługi autoryzacji

Ta funkcja jest udostępniana przez komponent usługi autoryzacji i jest wywoływana przez menedżer kolejek, gdy nie wymaga już usług tego komponentu. Funkcja musi wykonać wszystkie procedury czyszczące wymagane przez komponent.

Identyfikator funkcji dla tej funkcji (dla MQZEP) to MQZID_TERM_AUTHORITY.

Składnia

MQZ_TERM_AUTHORITY, *(Hconfig, Options, QMgrName, ComponentData,*
CompCode, Reason)

Parametry

Wywołanie MQZ_TERM_AUTHORITY ma następujące parametry.

Hconfig (MQHCONFIG)-wejście
Uchwyt konfiguracji.

Ten uchwyt reprezentuje konkretny komponent, który jest przerywany.

Opcje (MQLONG)-wejście
Opcje zakończenia.

Jest to jedna z poniższych nazw:

MQZTO_PRIMARY
Zakończenie podstawowe.

MQZTO_SECONDARY
Zakończenie drugorzędne.

QMgrName (MQCHAR48)-wejście
Nazwa menedżera kolejek.

Nazwa menedżera kolejek wywołującego komponent. Nazwa ta jest dopełniana spacjami do pełnej długości parametru; nazwa nie jest zakończona znakiem o kodzie zero.

Nazwa menedżera kolejek jest przekazywana do komponentu w celu uzyskania informacji. Interfejs usługi autoryzacji nie wymaga, aby komponent korzystał z niego w jakikolwiek zdefiniowany sposób.

ComponentData (MQBYTE x ComponentDataLength)-wejście/wyjście
Dane komponentu.

Te dane są przechowywane przez menedżer kolejek w imieniu tego konkretnego komponentu. Zmiany wprowadzone w tym komponencie przez dowolną z funkcji udostępnianych przez ten komponent są zachowywane i prezentowane przy następnym wywołaniu jednej z funkcji tego komponentu.

Długość tego obszaru danych jest przekazywana przez menedżer kolejek w parametrze **ComponentDataLength** wywołania MQZ_INIT_AUTHORITY.

Po zakończeniu wywołania MQZ_TERM_AUTHORITY menedżer kolejek usuwa te dane.

CompCode (MQLONG)-dane wyjściowe
Kod zakończenia.

Jest to jedna z poniższych nazw:

MQCC_OK
Zakończono pomyślnie.

MQCC_FAILED (niepowodzenie MQC)
Wywołanie nie powiodło się.

Przyczyna (MQLONG)-dane wyjściowe
Kod przyczyny określający *CompCode*.

Jeśli *CompCode* ma wartość MQCC_OK:

MQRC_BRAK
(0, X'000 ') Brak powodu do zgłoszenia.

Jeśli *CompCode* ma wartość MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE
(2285, X'8ED') Usługa bazowy jest niedostępna.

Niepowodzenie MQRC_TERMINATION_FAILED
(2287, X'8FF') Zakończenie nie powiodło się z niezdefiniowanej przyczyny.

Więcej informacji na temat tych kodów przyczyny zawiera sekcja Komunikaty i kody przyczyny.

Wywołanie C

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

Parametry przekazywane do usługi są deklarowane w następujący sposób:

```
MQHCONFIG  Hconfig;           /* Configuration handle */
MQLONG     Options;           /* Termination options */
MQCHAR48   QMgrName;          /* Queue manager name */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     CompCode;          /* Completion code */
MQLONG     Reason;            /* Reason code qualifying CompCode */
```

IBM i MQZAC (kontekst aplikacji) w systemie IBM i

Ten parametr określa dane związane z aplikacją wywołującą.

Struktura MQZAC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru **ApplicationContext** .

Pola

StrucId (MQCHAR4)
Identyfikator struktury.

Wartość jest następująca:

MQZAC_ID_STRUKTURY
Identyfikator struktury kontekstu aplikacji.

Dla języka programowania C zdefiniowana jest również stała MQZAC_STRUC_ID_ARRAY; ma ona taką samą wartość jak MQZAC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe dla usługi.

Wersja (MQLONG
Numer wersji struktury.

Wartość jest następująca:

MQZAC_VERSION_1
Struktura kontekstu aplikacji Version-1 .

Następująca stała określa numer wersji bieżącej:

MQZAC_CURRENT_VERSION
Bieżąca wersja struktury kontekstu aplikacji.

Jest to pole wejściowe dla usługi.

ProcessId (MQPID)
Identyfikator procesu.

Identyfikator procesu aplikacji.

ThreadId (MQTID)
Identyfikator wątku.

Identyfikator wątku aplikacji.

ApplName (MQCHAR28)
Nazwa aplikacji.

Nazwa aplikacji.

UserID (MQCHAR12)
Identyfikator użytkownika.

W systemach IBM i jest to profil użytkownika, w którym utworzono zadanie aplikacji. (W systemie IBM ipo przełączeniu profilu za pomocą funkcji API QWTSETP w zadaniu aplikacji zwracany jest bieżący profil użytkownika).

Identyfikator EffectiveUser(MQCHAR12)
Efektywny identyfikator użytkownika.

W systemach IBM i jest to bieżący profil użytkownika zadania aplikacji.

Środowisko (MQLONG)
Środowisko.

To pole określa środowisko, z którego wykonano wywołanie.

Może mieć jedną z następujących wartości:

SERWER MQXE_COMMAND_SERVER
Serwer komend.

MQXE_MQSC
Interpreter komend `runmqsc` .

Agent MQXE_MCA
Agent kanału komunikatów

MQXE_INNY
Niezdefiniowane środowisko

CallerType (MQLONG)
Typ programu wywołującego.

To pole określa typ programu, który wywołał wywołanie.

Może mieć jedną z następujących wartości:

MQXACT_EXTERNAL
Wywołanie jest zewnętrzne względem menedżera kolejek.

MQXACT_INTERNAL
Wywołanie jest wewnętrzne dla menedżera kolejek.

AuthenticationType (MQLONG)
Typ uwierzytelniania.

To pole określa typ wykonywanego uwierzytelniania.

Może mieć jedną z następujących wartości:

MQZAT_INITIAL_CONTEXT,
Wywołanie uwierzytelniania jest spowodowane zainicjowaniem kontekstu użytkownika. Ta wartość jest używana podczas wywołania `MQCONN` lub `MQCONNX` .

MQZAT_CHANGE_CONTEXT,
Wywołanie uwierzytelniania jest spowodowane zmianą kontekstu użytkownika. Ta wartość jest używana, gdy agent MCA zmienia kontekst użytkownika.

v

BindType (MQLONG
Typ powiązania.

To pole określa typ używanego powiązania.

Może mieć jedną z następujących wartości:

MQCNO_FASTPATH_BINDING,
Powiązanie krótkiej ścieżki.

MQCNO_SHARED_BINDING
Powiązanie współużytkowane.

MQCNO_IZOLOWANE_POWIĄZANIE
Izolowane powiązanie.

Deklaracja C

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
  MQCHAR4   StrucId;            /* Structure identifier */
  MQLONG    Version;            /* Structure version number */
  MQPID     ProcessId;          /* Process identifier */
  MQTID     ThreadId;           /* Thread identifier */
  MQCHAR28  ApplName;           /* Application name */
  MQCHAR12  UserID;             /* User identifier */
  MQCHAR12  EffectiveUserID;    /* Effective user identifier */
  MQLONG    Environment;        /* Environment */
  MQLONG    CallerType;         /* Caller type */
  MQLONG    AuthenticationType; /* Authentication type */
  MQLONG    BindType;           /* Bind type */
};
```

IBM i MQZAD (dane uprawnień) w systemie IBM i

Struktura MQZAD jest używana w wywołaniu MQZ_ENUMERATE_AUTHORITY_DATA dla dwóch parametrów.

Więcej informacji na temat parametrów **Filter** i **AuthorityBuffer** zawiera sekcja “MQZ_ENUMERATE_AUTHORITY_DATA (Enumerate authority data) w systemie IBM i” na stronie 1749 :

- Komenda MQZAD jest używana dla parametru **Filter** , który jest przekazywany do wywołania. Ten parametr określa kryteria wyboru, które mają być używane do wybierania danych uprawnień zwracanych przez wywołanie.
- Komenda MQZAD jest również używana dla parametru **AuthorityBuffer** , który jest wyprowadzany z wywołania. Ten parametr określa autoryzacje dla jednej kombinacji nazwy profilu, typu obiektu i jednostki.

Pola

StrucId (MQCHAR4)
Identyfikator struktury.

Wartość jest następująca:

ID_STRUKTURY_MQZAD_STRUCT
Identyfikator struktury danych uprawnień.

Dla języka programowania C zdefiniowana jest również stała MQZAD_STRUC_ID_ARRAY. Ma ona taką samą wartość jak MQZAD_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Jest to pole wejściowe dla usługi.

Wersja (MQLONG)
Numer wersji struktury.

Wartość jest następująca:

MQZAD_VERSION_1
Struktura danych uprawnień Version-1 .

Następująca stała określa numer wersji bieżącej:

MQZAD_CURRENT_VERSION
Bieżąca wersja struktury danych uprawnień.

Jest to pole wejściowe dla usługi.

ProfileName (MQCHAR48)
Nazwa profilu.

W przypadku parametru **Filter** w tym polu znajduje się nazwa profilu, z którego wymagane są dane uprawnień. Jeśli nazwa jest całkowicie pusta aż do końca pola lub pierwszego znaku o kodzie zero, zwracane są dane uprawnień dla wszystkich nazw profili.

Dla parametru **AuthorityBuffer** to pole jest nazwą profilu, który jest zgodny z podanymi kryteriami wyboru.

ObjectType (MQLONG)
Typ obiektu.

Dla parametru **Filter** to pole określa typ obiektu, dla którego wymagane są dane uprawnień. Jeśli wartością jest MQOT_ALL, zwracane są dane uprawnień dla wszystkich typów obiektów.

W przypadku parametru **AuthorityBuffer** pole to określa typ obiektu, do którego ma zastosowanie profil identyfikowany przez **ProfileName** .

Wartość jest jedną z następujących wartości; dla parametru **Filter** poprawna jest również wartość MQOT_ALL:

MQOT_AUTH_INFO
Informacje o uwierzytelnianiu.

MQOT_CHANNEL (kanał MQT)
Kanał.

MQOT_CLNTCONN_CHANNEL, kanał
Kanał połączenia klienta.

MQOT_LISTENER
Proces nasłuchujący.

LISTA NAZW MQOT_NAMELIST
Lista nazw.

PROCES_MQOT
Definicja procesu.

MQOT_Q
do kolejki błędów.

MQOT_Q_MGR
menedżerze kolejek.

USŁUACJA_MQOT
.

Uprawnienie (MQLONG)
Uprawnienia.

Dla parametru **Filter** to pole jest ignorowane.

Dla parametru **AuthorityBuffer** to pole reprezentuje autoryzacje, które jednostka ma do obiektów identyfikowanych przez **ProfileName** i **ObjectType**. Jeśli jednostka ma tylko jedno uprawnienie, pole jest równe odpowiedniej wartości autoryzacji (MQZAO_ * stała). Jeśli jednostka ma więcej niż jedno uprawnienie, pole jest bitową wartością OR odpowiednich stałych MQZAO_ *.

EntityDataPtr (PMQZED)
Adres struktury MQZED identyfikującej jednostkę.

Dla parametru **Filter** to pole wskazuje strukturę MQZED identyfikującą jednostkę, z której wymagane są dane uprawnień. Jeśli **EntityDataPtr** jest wskaźnikiem pustym, zwracane są dane uprawnień dla wszystkich jednostek.

Dla parametru **AuthorityBuffer** to pole wskazuje strukturę MQZED identyfikującą jednostkę, z której pochodzą zwracane dane uprawnień.

EntityType (MQLONG)
Typ jednostki.

W przypadku parametru **Filter** pole to określa typ jednostki, dla której wymagane są dane uprawnień. Jeśli wartością jest MQZAET_NONE, zwracane są dane uprawnień dla wszystkich typów jednostek.

W przypadku parametru **AuthorityBuffer** to pole określa typ jednostki identyfikowanej przez strukturę MQZED wskazaną przez parametr **EntityDataPtr**.

Jest to jedna z następujących wartości; w przypadku parametru **Filter** poprawna jest także wartość MQZAET_NONE:

MQZAET_PRINCIPAL
Podmiot kerberos.

GRUPA_MQZAET
Grupa.

Opcje (MQAUTHOPT)
Opcje.

To pole określa opcje, które dają kontrolę nad wyświetlanymi profilami.

Należy określić jedną z następujących wartości:

MQAUTHOPT_NAME_ALL_MATCHING
Wyświetla wszystkie profile

MQAUTHOPT_NAME_EXPLICIT
Wyświetla profile, które mają dokładnie taką samą nazwę, jak określona w polu **ProfileName** .

Ponadto należy również określić jedną z następujących opcji:

MQAUTHOPT_ENTITY_SET
Wyświetlenie wszystkich profili używanych do obliczenia skumulowanego uprawnienia jednostki do obiektu określonego przez **ProfileName**. Pole **ProfileName** nie może zawierać żadnych znaków wieloznacznych.

- Jeśli określona jednostka jest nazwą użytkownika, dla każdego elementu zestawu {entity, groups} wyświetlany jest najbardziej odpowiedni profil, który ma zastosowanie do obiektu.
- Jeśli określona jednostka jest grupą, wyświetlany jest najbardziej odpowiedni profil z grupy, który ma zastosowanie do obiektu.
- Jeśli ta wartość jest określona, wszystkie wartości parametrów **ProfileName**, **ObjectType**, **EntityType**i nazwa jednostki określone w strukturze **EntityDataPtr** MQZED muszą być niepuste.

Jeśli określono opcję *MQAUTHOPT_NAME_ALL_MATCHING*, można również określić następujące wartości:

MQAUTHOPT_ENTITY_EXPLICIT
Wyświetla profile, które mają dokładnie taką samą nazwę jednostki jak nazwa jednostki określona w strukturze **EntityDataPtr** MQZED.

Deklaracja C

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4   StrucId;        /* Structure identifier */
  MQLONG    Version;        /* Structure version number */
  MQCHAR48  ProfileName;    /* Profile name */
  MQLONG    ObjectType;     /* Object type */
  MQLONG    Authority;      /* Authority */
  PMQZED    EntityDataPtr;  /* Address of MQZED structure identifying an
```

```
                                entity */
  MQLONG    EntityType;     /* Entity type */
  MQAUTHOPT Options;        /* Options */
};
```

IBM i MQZED (deskryptor jednostki) w systemie IBM i

Struktura MQZED jest używana w wielu wywołaniach usługi autoryzacji w celu określenia jednostki, dla której ma zostać sprawdzona autoryzacja.

Pola

StrucId (MQCHAR4)
Identyfikator struktury.

Wartość jest następująca:

MQZED_STRUC_ID (identyfikator struktury MQ)
Identyfikator struktury deskryptora jednostki.

W języku programowania C zdefiniowana jest również stała MQZED_STRUC_ID_ARRAY, która ma taką samą wartość jak MQZED_STRUC_ID, ale jest tablicą znaków, a nie łańcuchem.

Jest to pole wejściowe dla usługi.

Wersja (MQLONG)
Numer wersji struktury.

Wartość jest następująca:

MQZED_VERSION_1
Version-1 struktura deskryptora jednostki.

Następująca stała określa numer wersji bieżącej:

MQZED_CURRENT_VERSION
Bieżąca wersja struktury deskryptora jednostki.

Jest to pole wejściowe dla usługi.

EntityNamePtr (PMQCHAR)
Adres nazwy jednostki.

Jest to wskaźnik do nazwy jednostki, której autoryzacja ma zostać sprawdzona.

EntityDomainPtr (PMQCHAR)
Adres nazwy domeny jednostki.

Jest to wskaźnik do nazwy domeny zawierającej definicję jednostki, której autoryzacja ma zostać sprawdzona.

SecurityId (MQBYTE40)
Identyfikator bezpieczeństwa.

Jest to identyfikator bezpieczeństwa, którego autoryzacja ma zostać sprawdzona.

CorrelationPtr (MQPTR)
Wskaźnik korelacji.

Ułatwia to przekazywanie danych korelacji między funkcją uwierzytelniania użytkownika i innymi odpowiednimi funkcjami OAM.

Deklaracja C

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
```

```
  PMQCHAR   EntityNamePtr;    /* Address of entity name */
  PMQCHAR   EntityDomainPtr;  /* Address of entity domain name */
  MQBYTE40  SecurityId;       /* Security identifier */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
```

IBM i MQZFP (parametry Free) w systemie IBM i

Ten parametr określa dane związane z zasobem, który ma zostać zwolniony.

Struktura MQZFP jest używana w wywołaniu MQZ_FREE_USER dla parametru **FreeParms** .

Pola

StrucId (MQCHAR4)
Identyfikator struktury.

Wartość jest następująca:

MQZFP_STRUC_ID (Identyfikator struktury MQZF)
Identyfikator struktury parametrów wolnych.

Dla języka programowania C zdefiniowana jest również stała MQZFP_STRUC_ID_ARRAY, która ma taką samą wartość jak MQZFP_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe dla usługi.

Wersja (MQLONG)
Numer wersji struktury.

Wartość jest następująca:

MQZFP_VERSION_1
Version-1 wolna struktura parametrów.

Następująca stała określa numer wersji bieżącej:

MQZFP_CURRENT_VERSION
Bieżąca wersja struktury wolnych parametrów.

Jest to pole wejściowe dla usługi.

Zarezerwowane (MQBYTE8)
Pole zastrzeżone.

Wartość początkowa jest pusta.

CorrelationPtr (MQPTR)
Wskaźnik korelacji.

Adres danych korelacji związanych z zasobem, który ma zostać zwolniony.

Deklaracja C

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQBYTE8   Reserved;         /* Reserved field */
  MQPTR     CorrelationPtr;   /* Address of correlation data */
};
```

IBM i MQZIC (kontekst tożsamości) w systemie IBM i

Struktura MQZIC jest używana w wywołaniu MQZ_AUTHENTICATE_USER dla parametru **IdentityContext** .

Struktura MQZIC zawiera informacje o kontekście tożsamości, które identyfikują użytkownika aplikacji, która najpierw umieściła komunikat w kolejce:

- Menedżer kolejek wypełnia pole UserIdentifier nazwą identyfikującą użytkownika. Sposób, w jaki menedżer kolejek może to zrobić, zależy od środowiska, w którym działa aplikacja.
- Menedżer kolejek wypełnia pole AccountingToken znacznikiem lub numerem określonym przez aplikację, która umieściła komunikat.
- Aplikacje mogą korzystać z pola danych ApplIdentityw celu uzyskania dodatkowych informacji o użytkowniku (na przykład zaszyfrowanego hasła).

Odpowiednio autoryzowane aplikacje mogą ustawić kontekst tożsamości za pomocą funkcji MQZ_AUTHENTICATE_USER.

Identyfikator bezpieczeństwa systemu Windows (SID) jest przechowywany w polu AccountingToken , gdy w systemie IBM MQ for Windowsjest tworzony komunikat. Identyfikatora SID można użyć do uzupełnienia pola UserIdentifier i do ustanowienia referencji użytkownika.

Pola

StrucId (MQCHAR4)
Identyfikator struktury.

Wartość jest następująca:

MQZIC_STRUC_ID (Identyfikator struktury MQ)
Identyfikator struktury kontekstu tożsamości.

Dla języka programowania C zdefiniowana jest również stała MQZIC_STRUC_ID_ARRAY, która ma taką samą wartość jak MQZIC_STRUC_ID, ale jest tablicą znaków zamiast łańcucha.

Jest to pole wejściowe dla usługi.

Wersja (MQLONG)
Numer wersji struktury.

Wartość jest następująca:

MQZIC_VERSION_1
Struktura kontekstu tożsamości Version-1 .

Następująca stała określa numer wersji bieżącej:

MQZIC_CURRENT_VERSION
Bieżąca wersja struktury kontekstu tożsamości.

Jest to pole wejściowe dla usługi.

UserIdentifier (MQCHAR12)
Identyfikator użytkownika.

Jest to część **kontekstu tożsamości** komunikatu.

UserIdentifier określa identyfikator użytkownika aplikacji, z której pochodzi komunikat. Menedżer kolejek traktuje te informacje jako dane znakowe, ale nie definiuje ich formatu. Więcej informacji na temat pola *UserIdentifier* zawiera sekcja "UserIdentifier (MQCHAR12) dla deskryptora MQMD" na stronie 470.

AccountingToken (MQBYTE32)
Token rozliczania.

Jest to część **kontekstu tożsamości** komunikatu.

AccountingToken umożliwia aplikacji spowodowanie odpowiedniego obciążenia pracą wykonaną w wyniku wysłania komunikatu. Menedżer kolejek traktuje te informacje jako łańcuch bitów i nie sprawdza ich treści. Więcej informacji na temat pola *AccountingToken* zawiera sekcja "AccountingToken (MQBYTE32) dla MQMD" na stronie 472.

Dane ApplIdentity(MQCHAR32)
Dane aplikacji odnoszące się do tożsamości.

Jest to część **kontekstu tożsamości** komunikatu.

ApplIdentityData to informacje zdefiniowane przez pakiet aplikacji, których można użyć do udostępnienia dodatkowych informacji o pochodzeniu komunikatu. Na przykład może być ona ustawiana przez aplikacje działające z odpowiednimi uprawnieniami użytkownika w celu wskazania, czy dane tożsamości są zaufane. Więcej informacji na temat pola *ApplIdentityData* zawiera sekcja “Dane ApplIdentity(MQCHAR32) dla MQMD” na stronie 473.

Deklaracja 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 */
};
```

Klasy i interfejsy IBM MQ .NET

Klasy i interfejsy IBM MQ .NET są wymienione w kolejności alfabetycznej. Opisano właściwości, metody i konstruktory.

Klasa MQAsyncStatus.NET

Aby uzyskać informacje o statusie poprzedniego działania MQI, na przykład o powodzeniu poprzednich asynchronicznych operacji put, należy użyć komendy MQAsyncStatus . MQAsyncStatus obudowuje funkcje struktury danych MQSTS .

Klasa

```
System.Object
    |
    └─ IBM.WMQ.MQBase
        |
        └─ IBM.WMQ.MQBaseObject
            |
            └─ IBM.WMQ.MQAsyncStatus
```

```
public class IBM.WMQ.MQAsyncStatus extends IBM.WMQ.MQBaseObject;
```

- “Właściwości” na stronie 1775
- “Konstruktory” na stronie 1776

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public static int CompCode {get;}

Kod zakończenia z pierwszego błędu lub ostrzeżenia.

public static int Reason {get;}

Kod przyczyny z pierwszego błędu lub ostrzeżenia.

public static int PutSuccessCount {get;}

Liczba pomyślnych asynchronicznych wywołań umieszczania MQI.

public static int PutWarningCount {get;}

Liczba wywołań asynchronicznych operacji umieszczania MQI, które zakończyły się powodzeniem, z ostrzeżeniem.

public static int PutFailureCount {get;}

Liczba nieudanych asynchronicznych wywołań umieszczania MQI.

public static int ObjectType {get;}

Typ obiektu dla pierwszego błędu. Dozwolone są następujące wartości:

- MQC.MQOT_ALIAS_Q
- MQC.MQOT_LOCAL_Q
- MQC.MQOT_MODEL_Q
- MQC.MQOT_Q
- MQC.MQOT_REMOTE_Q
- MQC.MQOT_TOPIC
- 0, co oznacza, że nie jest zwracany żaden obiekt

public static string ObjectName {get;}

Nazwa obiektu.

public static string ObjectQMgrName {get;}

Nazwa menedżera kolejek obiektu.

public static string ResolvedObjectName {get;}

Przetłumaczona nazwa obiektu.

public static string ResolvedObjectQMgrName {get;}

Nazwa menedżera kolejek rozstrzygniętego obiektu.

Konstruktory

public MQAsyncStatus() throws MQException;

Konstruktor metody, tworzy obiekt z polami inicjowanymi do zera lub do pustego pola.

Klasa MQAuthenticationInformationRecord.NET

Parametr MQAuthenticationInformationRecord umożliwia określenie informacji o elemencie uwierzytelniającym, który ma być używany w połączeniu klienta TLS produktu IBM MQ . MQAuthenticationInformationRecord hermetyzuje rekord informacji uwierzytelniającej, MQAIR.

Klasa

```
System.Object
    |
    └─ IBM.WMQ.MQAuthenticationInformationRecord
```

```
public class IBM.WMQ.MQAuthenticationInformationRecord extends System.Object;
```

- “Właściwości” na stronie 1776
- “Konstruktory” na stronie 1777

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public long Version {get; set;}

Numer wersji struktury.

public long AuthInfoType {get; set;}

Typ informacji uwierzytelniającej. Ten atrybut musi mieć jedną z następujących wartości:

- OCSP -sprawdzanie statusu odwołania certyfikatu jest wykonywane przy użyciu protokołu OCSP.
- CRLLDAP -Sprawdzanie statusu odwołania certyfikatu jest wykonywane przy użyciu list odwołań certyfikatów na serwerach LDAP.

public string AuthInfoConnName {get; set;}

Nazwa DNS lub adres IP hosta, na którym działa serwer LDAP, z opcjonalnym numerem portu. To słowo kluczowe jest wymagane.

public string LDAPPassword {get; set;}

Hasło powiązane z nazwą wyróżniającą użytkownika uzyskującego dostęp do serwera LDAP. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość **AuthInfoType** ma wartość CRLLDAP.

public string LDAPUserName {get; set;}

Nazwa wyróżniająca użytkownika uzyskującego dostęp do serwera LDAP. Po ustawieniu tej właściwości `LDAPUserNameLength` i `LDAPUserNamePtr` są automatycznie ustawiane poprawnie. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość `AuthInfoType` ma wartość CRLLDAP.

public string OCSPResponderURL {get; set;}

Adres URL, przy użyciu którego można nawiązać połączenie z modułem odpowiadającym OCSP. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość `AuthInfoType` ma wartość OCSP .

W tym polu rozróżniana jest wielkość liter. Musi zaczynać się od łańcucha http:// pisanego małymi literami. W pozostałych URL może być rozróżniana wielkość liter, w zależności od implementacji serwera OCSP.

Konstruktory

MQAuthenticationInformationRecord();

Klasa `MQDestination.NET`

Aby uzyskać dostęp do metod, które są wspólne dla produktów `MQQueue` i `MQTopic`, należy użyć metody `MQDestination` . `MQDestination` jest abstrakcyjną klasą bazową i nie można utworzyć jej instancji.

Klasa

```
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;`

- “Właściwości” na stronie 1778
- “metody” na stronie 1778

- “Konstruktory” na stronie 1780

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public DateTime CreationDateTime {get;}

Data i godzina utworzenia kolejki lub tematu. Oryginalnie zawarta w pliku MQQueue, ta właściwość została przeniesiona do klasy bazowej MQDestination .

Nie istnieje wartość domyślna.

public int DestinationType {get;}

Liczba całkowita opisująca typ używanego miejsca docelowego. Zainicjowana z konstruktora podklas MQQueue lub MQTopic, ta wartość może przyjmować jedną z następujących wartości:

- MQOT_Q
- MQOT_TOPIC

Nie istnieje wartość domyślna.

metody

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Zwraca wyjątek MQException.

Pobiera komunikat z kolejki, jeśli miejsce docelowe jest obiektem MQQueue , lub z tematu, jeśli miejsce docelowe jest obiektem MQTopic , używając domyślnej instancji MQGetMessageOptions do wykonania pobierania.

Jeśli operacja pobierania nie powiedzie się, obiekt MQMessage pozostanie niezmieniony. Jeśli operacja powiedzie się, części deskryptora komunikatu i danych komunikatu w pliku MQMessage są zastępowane deskryptorem komunikatu i danymi komunikatu z komunikatu przychodzącego.

Wszystkie wywołania IBM MQ z określonego MQQueueManager są synchroniczne. Z tego powodu, jeśli wykonywana jest operacja get with wait, wszystkie inne wątki korzystające z tego samego MQQueueManager będą miały zablokowaną możliwość wykonywania kolejnych wywołań IBM MQ do czasu wykonania wywołania Get. Jeśli dostęp do programu IBM MQ ma być uzyskiwany jednocześnie z wielu wątków, każdy wątek musi utworzyć własny obiekt MQQueueManager .

komunikat
Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskryptorze komunikatu są parametrami wejściowymi. Należy upewnić się, że parametry wejściowe MessageId i CorrelationId są ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia dla komunikatów odebranych w ramach elementu MQGM_SYNCPOINT.

Opcje getMessage
Opcje sterujące działaniem pobierania.

Użycie opcji MQC.MQGMO_CONVERT może spowodować wystąpienie wyjątku z kodem przyczyny MQC.MQRC_CONVERTED_STRING_TOO_BIG podczas przekształcania kodów znaków jednobajtowych w kody dwubajtowe. W takim przypadku komunikat jest kopiowany do buforu bez konwersji.

Jeśli parametr *getMessageOptions* nie jest określony, używana jest opcja komunikatu MQGMO_NOWAIT.

Jeśli opcja MQGMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

MaxMsg
Największy komunikat, który ma zostać odebrany przez ten obiekt komunikatu. Jeśli komunikat w kolejce jest większy niż ta wielkość, wystąpi jedna z dwóch sytuacji:

- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG jest ustawiona w obiekcie MQGetMessageOptions , komunikat jest wypełniany możliwie największą ilością danych komunikatu. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_ACCEPTED .
- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie jest ustawiona, komunikat pozostaje w kolejce. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_FAILED .

Jeśli parametr *MaxMsgSize* nie jest określony, pobierany jest cały komunikat.

```
public void Put(MQMessage message);
public void Put(MQMessage message, MQPutMessageOptions putMessageOptions);
```

Zwraca wyjątek MQException.

Umieszcza komunikat w kolejce, jeśli miejsce docelowe jest obiektem MQQueue , lub publikuje komunikat w temacie, jeśli miejsce docelowe jest obiektem MQTopic .

Modyfikacje obiektu MQMessage dokonane po wywołaniu metody Put nie mają wpływu na rzeczywisty komunikat w kolejce produktu IBM MQ lub w temacie publikacji.

Put aktualizuje właściwości MessageId i CorrelationId obiektu MQMessage i nie czyści danych komunikatu. Dalsze wywołania Put lub Get odnoszą się do zaktualizowanych informacji w obiekcie MQMessage . Na przykład w poniższym fragmencie kodu pierwszy komunikat zawiera a , a drugi ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

komunikat
Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat do wysłania. W wyniku tej metody można zmienić deskryptor komunikatu. Wartości w deskryptorze komunikatu bezpośrednio po zakończeniu tej metody są wartościami, które zostały umieszczone w kolejce lub opublikowane w temacie.

Do klienta z możliwością ponownego połączenia zwracane są następujące kody przyczyny:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało zerwane podczas wykonywania wywołania metody Put dla trwałego komunikatu i ponowne połączenie zakończyło się pomyślnie.
- MQRC_NONE , jeśli połączenie zostało pomyślnie nawiązane podczas wykonywania wywołania metody Put dla komunikatu nietrwałego (patrz sekcja Odtwarzanie aplikacji).

Opcje komendy putMessage
Opcje sterujące działaniem operacji put.

Jeśli parametr *putMessageOptions* nie zostanie podany, zostanie użyta domyślna instancja MQPutMessageOptons .

Jeśli opcja MQPMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

Uwaga: W celu uproszczenia i zwiększenia wydajności, jeśli pojedynczy komunikat ma zostać umieszczony w kolejce, należy użyć obiektu MQQueueManager.Put . W tym celu powinien istnieć obiekt MQQueue .

Konstruktory

`MQDestination` jest abstrakcyjną klasą bazową i nie można utworzyć jej instancji. Uzyskaj dostęp do miejsc docelowych przy użyciu konstruktorów `MQQueue` i `MQTopic` lub przy użyciu konstruktorów `MQQueueManager.AccessQueue` i `MQQueueManager.AccessTopic methods`.

Klasa MQEnvironment.NET

Parametr `MQEnvironment` umożliwia sterowanie sposobem wywołania konstruktora `MQQueueManager` i wybór połączenia IBM MQ MQI client . Klasa `MQEnvironment` zawiera właściwości sterujące zachowaniem klasy IBM MQ.

Klasa

```
System.Object
        |
        └ IBM.WMQ.MQEnvironment
```

`public class IBM.WMQ.MQEnvironment extends System.Object;`

- "Właściwości-tylko klient" na stronie 1780
- "Właściwości" na stronie 1781
- "Konstruktory" na stronie 1782

Właściwości-tylko klient

Test `MQException` zgłaszany podczas pobierania właściwości.

`public static int CertificateValPolicy {get; set;}`

Określa, która strategia sprawdzania poprawności certyfikatu TLS jest używana do sprawdzania poprawności certyfikatów cyfrowych odebranych ze zdalnych systemów partnerskich. Poprawne wartości:

- MQC.CERTIFICATE_VALIDATION_POLICY_ANY
- MQC.CERTIFICATE_VALIDATION_POLICY_RFC5280

`public static ArrayList EncryptionPolicySuiteB {get; set;}`

Ustawia poziom kryptografii zgodnej z pakietem Suite B. Poprawne wartości:

- MQC.MQ_SUITE_B_NONE -wartość domyślna.
- MQC.MQ_SUITE_B_128_BIT
- MQC.MQ_SUITE_B_192_BIT

`public static string Channel {get; set;}`

Nazwa kanału, z którym ma zostać nawiązane połączenie z docelowym menedżerem kolejek. Przed utworzeniem instancji `MQQueueManager` w trybie klienta należy ustawić właściwość kanału.

`public static int FipsRequired {get; set;}`

Należy podać wartość `MQC.MQSSL_FIPS_YES` , aby używać tylko algorytmów z certyfikatem FIPS, jeśli szyfrowanie jest wykonywane w produkcie IBM MQ. Wartością domyślną jest `MQC.MQSSL_FIPS_NO`.

Jeśli sprzęt szyfrujący jest skonfigurowany, używane są moduły szyfrujące dostarczane przez produkt sprzętowy. W zależności od używanego sprzętu mogą one nie mieć certyfikatu FIPS na określonym poziomie.

public static string Hostname {get; set;}
Nazwa hosta TCP/IP komputera, na którym znajduje się serwer IBM MQ . Jeśli nazwa hosta nie jest ustawiona i nie są ustawione żadne właściwości przesłaniające, do nawiązywania połączenia z lokalnym menedżerem kolejek używany jest tryb powiązań serwera.

public static int Port {get; set;}
Port, z którym ma zostać nawiązane połączenie. Jest to port, na którym serwer IBM MQ nasłuchuje żądań połączeń przychodzących. Wartością domyślną jest 1414.

public static string SSLCipherSpec {get; set;}
Ustaw wartość parametru SSLCipherSpec na wartość CipherSpec ustawioną w kanale SVRCONN, aby włączyć obsługę protokołu TLS dla połączenia. Wartością domyślną jest NULL, a protokół TLS nie jest włączony dla połączenia.

public static string sslPeerName {get; set;}

Wzorzec nazwy wyróżniającej. Jeśli parametr sslCipherSpec jest ustawiony, można użyć tej zmiennej, aby upewnić się, że używany jest poprawny menedżer kolejek. Jeśli ma wartość null (wartość domyślna), nazwa wyróżniająca (DN) menedżera kolejek nie jest wykonywana. sslPeerNazwa jest ignorowana, jeśli sslCipherSpec ma wartość null.

V 9.3.0 V 9.3.0 **public static string SSLKeyRepository {get; set;}**
Jawny tekst lub zaszyfrowane hasło dostępu do repozytorium kluczy. Frazy hasła repozytorium kluczy są szyfrowane do użycia przez aplikacje klienckie za pomocą programu narzędziowego **runmqicred** .

Jeśli parametr SSLKeyRepositoryPassword ma wartość NULL (wartość domyślna), używana jest wartość zmiennej środowiskowej *MQKEYRPWD* lub atrybutu **SSLKeyRepositoryPassword** w pliku konfiguracyjnym klienta.

V 9.3.0 V 9.3.0 **public static string InitialKey {get; set;}**
Klucz początkowy, który został użyty do zaszyfrowania frazy hasła repozytorium kluczy określonej w parametrze SSLKeyRepositoryPassword.

Klucz początkowy musi być określony, jeśli początkowy plik kluczy został określony podczas szyfrowania frazy hasła repozytorium kluczy za pomocą programu narzędziowego **runmqicred** .

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public static ArrayList HdrCompList {get; set;}
Lista kompresji danych nagłówka

public static int KeyResetCount {get; set;}
Wskazuje liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji TLS przed renegocjacją klucza tajnego.

public static ArrayList MQAIRArray {get; set;}
Tablica obiektów MQAuthenticationInformationRecord .

public static ArrayList MsgCompList {get; set;}
Lista kompresji danych komunikatu

public static string Password {get; set;}
Hasło do uwierzytelnienia. Hasło, do którego odwołuje się struktura MQCSP , jest zapełniane przez ustawienie tej właściwości Hasło.

public static string ReceiveExit {get; set;}
Wyjście odbierania umożliwia sprawdzanie i modyfikowanie danych odebranych z menedżera kolejek. Zwykle jest on używany z odpowiednim wyjściem wysyłania w menedżerze kolejek. Jeśli parametr ReceiveExit ma wartość null, nie jest wywoływane żadne wyjście receive.

public static string ReceiveUserData {get; set;}
Dane użytkownika powiązane z wyjściem odbierania. Ograniczona do 32 znaków.

public static string SecurityExit {get; set;}
Wyjście zabezpieczeń umożliwia dostosowanie przepływów zabezpieczeń, które występują podczas próby nawiązania połączenia z menedżerem kolejek. Jeśli parametr `SecurityExit` ma wartość null, nie jest wywoływane żadne wyjście zabezpieczeń.

public static string SecurityUserData {get; set;}
Dane użytkownika powiązane z wyjściem zabezpieczeń. Ograniczona do 32 znaków.

public static string SendExit {get; set;}
Wyjście wysyłania umożliwia sprawdzenie lub zmianę danych wysyłanych do menedżera kolejek. Zwykle jest on używany z odpowiednim wyjściem odbierania w menedżerze kolejek. Jeśli parametr `SendExit` ma wartość null, nie jest wywoływane żadne wyjście wysyłania.

public static string SendUserData {get; set;}
Dane użytkownika powiązane z wyjściem wysyłania. Ograniczona do 32 znaków.

public static string SharingConversations {get; set;}

Pole `SharingConversations` jest używane w połączeniach z aplikacji .NET , jeśli te aplikacje nie używają tabeli definicji kanału klienta (CCDT).

`SharingConversations` określa maksymalną liczbę konwersacji, które mogą być współużytkowane w gnieździe powiązanym z tym połączeniem.

Wartość 0 oznacza, że kanał działa tak, jak przed IBM WebSphere MQ 7.0, w odniesieniu do współużytkowania konwersacji, odczytu z wyprzedzeniem i pulsu.

Pole jest przekazywane w tabeli mieszającej właściwości jako `SHARING_CONVERSATIONS_PROPERTY`podczas tworzenia instancji menedżera kolejek produktu IBM MQ .

Jeśli opcja `SharingConversations`nie zostanie podana, zostanie użyta wartość domyślna 10.

public static string SSLCryptoHardware {get; set;}

Ustawia nazwę łańcucha parametru wymaganego do skonfigurowania sprzętu szyfrującego w systemie. Parametr `SSLCryptoHardware` jest ignorowany, jeśli parametr `sslCipherSpec` ma wartość null.

public static string SSLKeyRepository {get; set;}

Ustaw pełną nazwę pliku repozytorium kluczy.

V 9.3.0 V 9.3.0 Jeśli rozszerzenie nazwy pliku nie zostanie podane, przyjmuje się, że jest to rozszerzenie `.kdb` .

Jeśli parametr `SSLKeyRepository` ma wartość NULL (wartość domyślna), do znalezienia repozytorium kluczy używana jest zmienna środowiskowa `MQSSLKEYR` certyfikatu.

public static string UserId {get; set;}

Identyfikator użytkownika, który ma zostać uwierzytelniony. Identyfikator użytkownika, do którego odwołuje się struktura `MQCSP` , jest zapełniany przez ustawienie `UserId`. Uwierzytelnij użytkownika `UserId` , używając funkcji API lub wyjścia zabezpieczeń.

Konstruktory

public MQEnvironment()

Klasa MQException.NET

Aby znaleźć kod zakończenia i kod przyczyny funkcji IBM MQ zakończonej niepowodzeniem, należy użyć komendy `MQException` . Wyjątek `MQException` jest zgłaszany za każdym razem, gdy wystąpi błąd IBM MQ .

Klasa

```
System.Object
        |
        └─ System.Exception
                |
                └─ System.ApplicationException
                        |
                        └─ IBM.WMQ.MQException
```

```
public class IBM.WMQ.MQException extends System.ApplicationException;
```

- "Właściwości" na stronie 1783
- "Konstruktory" na stronie 1783

Właściwości

public int CompletionCode {get; set;}
Kod zakończenia IBM MQ powiązany z błędem. Możliwe wartości:

- `MQException.MQCC_OK`
- `MQException.MQCC_WARNING`
- `MQException.MQCC_FAILED`

public int ReasonCode {get; set;}
Kod przyczyny IBM MQ opisujący błąd.

Konstruktory

public MQException(int *completionCode*, int *reasonCode*)

completionCode
Kod zakończenia IBM MQ .

reasonCode
Kod zakończenia IBM MQ .

Klasa MQGetMessageOptions.NET

Parametr `MQGetMessageOptions` umożliwia określenie sposobu pobierania komunikatów. Modyfikuje on zachowanie `MQDestination.Get`.

Klasa

```
System.Object
        |
        └─ IBM.WMQ.MQBase
                |
                └─ IBM.WMQ.MQBaseObject
                        |
                        └─ IBM.WMQ.MQGetMessageOptions
```

```
public class IBM.WMQ.MQGetMessageOptions extends IBM.WMQ.MQBaseObject;
```

- "Właściwości" na stronie 1783
- "Konstruktory" na stronie 1786

Właściwości

Uwaga: Zachowanie niektórych opcji dostępnych w tej klasie zależy od środowiska, w którym są używane. Te elementy są oznaczone gwiazdką *.

Test MQException zgłaszany podczas pobierania właściwości.

public int GroupStatus {get;}*
GroupStatus wskazuje, czy pobrany komunikat jest w grupie i czy jest ostatnim komunikatem w grupie. Dozwolone są następujące wartości:

MQC.MQGS_LAST_MSG_IN_GROUP
Komunikat jest ostatnim lub jedynym komunikatem w grupie.

MQC.MQGS_MSG_IN_GROUP
Komunikat znajduje się w grupie, ale nie jest ostatnim w grupie.

MQC.MQGS_NOT_IN_GROUP
Komunikat nie należy do grupy.

public int MatchOptions {get; set;}*
MatchOptions określa sposób wybierania komunikatu. Można ustawić następujące opcje dopasowania:

MQC.MQMO_MATCH_CORREL_ID
Identyfikator korelacji do dopasowania.

MQC.MQMO_MATCH_GROUP_ID
Identyfikator grupy do dopasowania.

MQC.MQMO_MATCH_MSG_ID
Identyfikator komunikatu do dopasowania.

MQC.MQMO_MATCH_MSG_SEQ_NUMBER
Dopasuj numer kolejny komunikatu.

MQC.MQMO_NONE
Dopasowanie nie jest wymagane.

public int Options {get; set;}
Opcje sterują działaniem MQQueue.get. Można podać dowolną z następujących wartości. Jeśli wymagana jest więcej niż jedna opcja, wartości można dodać lub połączyć za pomocą operatora OR.

MQC.MQGMO_ACCEPT_TRUNCATED_MSG
Zezwalaj na obcięcie danych komunikatu.

MQC.MQGMO_ALL_MSGS_AVAILABLE*
Pobierz komunikaty z grupy tylko wtedy, gdy wszystkie komunikaty w grupie są dostępne.

MQC.MQGMO_ALL_SEGMENTS_AVAILABLE*
Pobierz segmenty komunikatu logicznego tylko wtedy, gdy wszystkie segmenty w grupie są dostępne.

MQC.MQGMO_BROWSE_FIRST
Przeglądaj od początku kolejki.

MQC.MQGMO_BROWSE_MSG_UNDER_CURSOR*
Komunikat przeglądania pod kursorem przeglądania.

MQC.MQGMO_BROWSE_NEXT
Przeglądaj z bieżącej pozycji w kolejce.

MQC.MQGMO_COMPLETE_MSG*
Pobierz tylko kompletne komunikaty logiczne.

MQC.MQGMO_CONVERT
Przed skopiowaniem danych do buforu komunikatów należy zażądać konwersji danych aplikacji w celu zapewnienia zgodności z atrybutami CharacterSet i Encoding MQMessage. Ponieważ konwersja danych jest również stosowana, gdy dane są pobierane z buforu komunikatów, aplikacje nie ustawiają tej opcji.

Użycie tej opcji może spowodować problemy podczas konwersji zestawów znaków jednobajtowych na zestawy znaków dwubajtowych. Zamiast tego należy wykonać konwersję przy użyciu metod readString, readLinei writeString po dostarczeniu komunikatu.

MQC.MQGMO_FAIL_IF_QUIESCING
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

MQC.MQGMO_LOCK*
Zablokuj komunikat, który jest przeglądany.

MQC.MQGMO_LOGICAL_ORDER*
Zwraca komunikaty w grupach i segmentach komunikatów logicznych w kolejności logicznej.

Jeśli opcja MQGMO_LOGICAL_ORDER zostanie użyta w kliencie z możliwością ponownego połączenia, kod przyczyny MQRC_RECONNECT_INCOMPATIBLE zostanie zwrócony do aplikacji.

MQC.MQGMO_MARK_SKIP_BACKOUT*
Zezwalaj na wycofywanie jednostki pracy bez przywracania komunikatu do kolejki.

MQC.MQGMO_MSG_UNDER_CURSOR
Pobierz komunikat pod kursorem przeglądania.

MQC.MQGMO_NONE
Nie określono żadnych innych opcji; wszystkie opcje przyjmują wartości domyślne.

MQC.MQGMO_NO_PROPERTIES
Nie są pobierane żadne właściwości komunikatu, z wyjątkiem właściwości zawartych w deskryptorze komunikatu (lub rozszerzeniu).

MQC.MQGMO_NO_SYNCPOINT
Pobierz komunikat bez sterowania punktem synchronizacji.

MQC.MQGMO_NO_WAIT
Wróć natychmiast, jeśli nie ma odpowiedniego komunikatu.

MQC.MQGMO_PROPERTIES_AS_Q_DEF
Pobieranie właściwości komunikatu zgodnie z definicją atrybutu PropertyControl elementu MQQueue. Atrybut PropertyControl nie ma wpływu na dostęp do właściwości komunikatu w deskryptorze lub rozszerzeniu komunikatu.

MQC.MQGMO_PROPERTIES_COMPATIBILITY
Pobierz właściwości komunikatu z przedrostkiem mcd, jms, usrlub mqextw nagłówkach MQRFH2 . Inne właściwości komunikatu, z wyjątkiem właściwości zawartych w deskryptorze komunikatu lub rozszerzeniu, są odrzucane.

MQC.MQGMO_PROPERTIES_FORCE_MQRFH2
Pobieranie właściwości komunikatu, z wyjątkiem właściwości zawartych w deskryptorze komunikatu lub rozszerzeniu w nagłówkach MQRFH2 . Należy użyć wartości MQC.MQGMO_PROPERTIES_FORCE_MQRFH2 w aplikacjach, które oczekują na pobranie właściwości, ale nie można jej zmienić, aby używały uchwytów komunikatów.

MQC.MQGMO_PROPERTIES_IN_HANDLE
Pobranie właściwości komunikatu przy użyciu uchwytu MsgHandle.

MQC.MQGMO_SYNCPOINT
Pobierz komunikat pod kontrolą punktu synchronizacji. Komunikat jest oznaczany jako niedostępny dla innych aplikacji, ale jest usuwany z kolejki tylko wtedy, gdy jednostka pracy jest zatwierdzona. Komunikat zostanie ponownie udostępniony, jeśli jednostka pracy zostanie wycofana.

MQC.MQGMO_SYNCPOINT_IF_PERSISTENT*
Pobierz komunikat z elementem sterującym punktu synchronizacji, jeśli komunikat jest trwały.

MQC.MQGMO_UNLOCK*
Odblokuj poprzednio zablokowany komunikat.

MQC.MQGMO_WAIT
Poczekaj na nadejście komunikatu.

public string ResolvedQueueName {get;}
Menedżer kolejek ustawia wartość ResolvedQueueNazwa na nazwę lokalną kolejki, z której został pobrany komunikat. ResolvedQueueNazwa różni się od nazwy użytej do otwarcia kolejki, jeśli kolejka aliasowa lub kolejka modelowa została otwarta.

public char Segmentation {get;}*
Segmentacja wskazuje, czy można zezwolić na segmentację dla pobranego komunikatu. Dozwolone są następujące wartości:

MQC.MQSEG_INHIBITED
Nie zezwalaj na segmentację.

MQC.MQSEG_ALLOWED
Zezwalaj na segmentację

public byte SegmentStatus {get;}*
SegmentStatus to pole wyjściowe wskazujące, czy pobrany komunikat jest segmentem komunikatu logicznego. Jeśli komunikat jest segmentem, flaga wskazuje, czy jest to ostatni segment. Dozwolone są następujące wartości:

MQC.MQSS_LAST_SEGMENT
Komunikat jest ostatnim lub jedynym segmentem komunikatu logicznego.

MQC.MQSS_NOT_A_SEGMENT
Komunikat nie jest segmentem.

MQC.MQSS_SEGMENT
Komunikat jest segmentem, ale nie jest ostatnim segmentem komunikatu logicznego.

public int WaitInterval {get; set;}
WaitInterval to maksymalny czas w milisekundach, przez który wywołanie MQQueue.get oczekuje na odebranie odpowiedniego komunikatu. Użyj opcji WaitInterval z MQC.MQGMO_WAIT. Ustaw wartość MQC.MQWI_UNLIMITED , aby odczekać nieograniczony czas na komunikat.

Konstruktory

public MQGetMessageOptions()
Utwórz nowy obiekt MQGetMessageOptions z opcją Options ustawioną na wartość MQC.MQGMO_NO_WAIT, opcją WaitInterval ustawioną na wartość zero i wartością ResolvedQueueName ustawioną na wartość pustą.

Klasa MQManagedObject.NET

Użyj MQManagedObject , aby uzyskać i ustawić atrybuty MQDestination, MQProcess, MQQueueManageri MQSubscription. MQManagedObject jest nadklasą tych klas.

Klasy

```
System.Object
        |
        └─ IBM.WMQ.MQBase
                |
                └─ IBM.WMQ.MQBaseObject
                        |
                        └─ IBM.WMQ.MQManagedObject
                                |
                                └─ IBM.WMQ.MQProcess
                                |
                                └─ IBM.WMQ.MQQueueManager
                                |
                                └─ IBM.WMQ.MQSubscription
                                |
                                └─ IBM.WMQ.MQDestination
```

public class IBM.WMQ.MQManagedObject extends IBM.WMQ.MQBaseObject;

- “Właściwości” na stronie 1787
- “metody” na stronie 1787
- “Konstruktory” na stronie 1789

Właściwości

Test `MQException` zgłaszany podczas pobierania właściwości.

public string AlternateUserId {get; set;}
Alternatywny identyfikator użytkownika (jeśli istnieje) ustawiony podczas otwierania zasobu. `AlternateUserID.set` jest ignorowany w przypadku wydawania dla obiektu, który jest otwarty. Identyfikator `AlternateUserId` nie jest poprawny dla subskrypcji.

public int CloseOptions {get; set;}

Ustaw ten atrybut, aby kontrolować sposób zamykania zasobu. Wartością domyślną jest `MQC.MQCO_NONE`. `MQC.MQCO_NONE` to jedyna dozwolona wartość dla wszystkich zasobów innych niż trwałe kolejki dynamiczne, tymczasowe kolejki dynamiczne, subskrypcje i tematy, do których obiekty, które je utworzyły, uzyskują dostęp.

W przypadku kolejek i tematów dozwolone są następujące wartości dodatkowe:

MQC.MQCO_DELETE
Usuń kolejkę, jeśli nie ma żadnych komunikatów.

MQC.MQCO_DELETE_PURGE
Usuń kolejkę, usuwając wszystkie znajdujące się w niej komunikaty.

MQC.MQCO_QUIESCE
Zażądaj zamknięcia kolejki, otrzymując ostrzeżenie, jeśli jakiekolwiek komunikaty pozostaną (co umożliwi ich pobranie przed zamknięciem końcowym).

W przypadku subskrypcji dozwolone są następujące wartości dodatkowe:

MQC.MQCO_KEEP_SUB
Subskrypcja nie została usunięta. Ta opcja jest poprawna tylko wtedy, gdy pierwotna subskrypcja jest trwała. `MQC.MQCO_KEEP_SUB` jest wartością domyślną dla tematu trwałego.

MQC.MQCO_REMOVE_SUB
Subskrypcja została usunięta. `MQC.MQCO_REMOVE_SUB` jest wartością domyślną dla nietrwałego, niezarządzanego tematu.

MQC.MQCO_PURGE_SUB
Subskrypcja została usunięta. `MQC.MQCO_PURGE_SUB` jest wartością domyślną dla nietrwałego, zarządzanego tematu.

public MQQueueManager ConnectionReference {get;}
Menedżer kolejek, do którego należy ten zasób.

public string MQDescription {get;}
Opis zasobu wstrzymany przez menedżer kolejek. `MQDescription` zwraca pusty łańcuch dla subskrypcji i tematów.

public boolean IsOpen {get;}
Wskazuje, czy zasób jest obecnie otwarty.

public string Name {get;}
Nazwa zasobu. Jest to nazwa podana w metodzie dostępu lub nazwa przydzielona przez menedżera kolejek dla kolejki dynamicznej.

public int OpenOptions {get; set;}
OpenOptions są ustawiane podczas otwierania obiektu IBM MQ . Metoda `OpenOptions.set` jest ignorowana i nie powoduje błędu. Subskrypcje nie mają `OpenOptions`.

metody

public virtual void Close();
Zwraca wyjątek `MQException`.

Zamyka obiekt. Po wywołaniu metody `Close`nie są dozwolone żadne dalsze operacje na tym zasobie. Aby zmienić zachowanie metody `Close` , należy ustawić atrybut `closeOptions` .

public string GetAttributeString(int *selector*, int *length*);
Zwraca wyjątek MQException.

Pobiera łańcuch atrybutu.

selektor
Liczba całkowita wskazująca, który atrybut jest odpytywany.

długość
Liczba całkowita wskazująca wymaganą długość łańcucha.

public void Inquire(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Zwraca wyjątek MQException.

Zwraca tablicę liczb całkowitych i zestaw łańcuchów znaków zawierający atrybuty kolejki, procesu lub menedżera kolejek. Atrybuty, które mają być odpytywane, są określone w tablicy selektorów.

Uwaga: Wiele z bardziej powszechnych atrybutów może być odpytywanych za pomocą metod Get zdefiniowanych w MQManagedObject, MQQueue i MQQueueManager.

selektory
Tablica liczb całkowitych identyfikująca atrybuty z wartościami, których dotyczy zapytanie.

intAttrs
Tablica, w której zwracane są wartości atrybutów będące liczbami całkowitymi. Wartości atrybutów całkowitych są zwracane w tej samej kolejności, co selektory atrybutów całkowitych w tablicy selektorów.

charAttrs
Bufor, w którym są zwracane atrybuty znakowe, skonkatenowany. Atrybuty znakowe są zwracane w tej samej kolejności, co selektory atrybutów znakowych w tablicy selektorów. Długość każdego łańcucha atrybutu jest ustalona dla każdego atrybutu.

public void Set(int[] *selectors*, int[] *intAttrs*, byte[] *charAttrs*);
Zwraca wyjątek MQException.

Ustawia atrybuty zdefiniowane w wektorze selektorów. Atrybuty, które mają zostać ustawione, są określone w tablicy selektorów.

selektory
Tablica liczb całkowitych identyfikująca atrybuty z wartościami do ustawienia.

intAttrs
Tablica wartości atrybutów całkowitych, które mają zostać ustawione. Wartości te muszą być w tej samej kolejności, co selektory atrybutów będące liczbami całkowitymi w tablicy selektorów.

charAttrs
Bufor, w którym są konkatenowane atrybuty znaków do ustawienia. Wartości te muszą być w tej samej kolejności, co selektory atrybutów znakowych w tablicy selektorów. Długość każdego atrybutu znakowego jest stała.

public void SetAttributeString(int *selector*, string *value*, int *length*);
Zwraca wyjątek MQException.

Ustawia łańcuch atrybutu.

selektor
Liczba całkowita wskazująca, który atrybut jest ustawiany.

wartość
Łańcuch, który ma zostać ustawiony jako wartość atrybutu.

długość
Liczba całkowita wskazująca wymaganą długość łańcucha.

Konstruktory

protected MQManagedObject()
Metoda konstruktora. Ten obiekt jest abstrakcyjną klasą bazową, której nie można utworzyć samodzielnie.

Klasa MQMessage.NET

Aby uzyskać dostęp do deskryptora komunikatu i danych dla komunikatu IBM MQ , należy użyć parametru MQMessage . MQMessage hermetyzuje komunikat IBM MQ .

Klasa

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.MQMessage
```

```
public class IBM.WMQ.MQMessage extends IBM.WMQ.MQBaseObject;
```

Utwórz obiekt MQMessage , a następnie użyj metod Read i Write , aby przesłać dane między komunikatem i innymi obiektami w aplikacji. Wysyłanie i odbieranie obiektów MQMessage za pomocą metod Put i Get klas MQDestination, MQQueue i MQTopic .

Pobierz i ustaw właściwości deskryptora komunikatu przy użyciu właściwości MQMessage. Ustawianie i pobieranie rozszerzonych właściwości komunikatu przy użyciu metod SetProperty i GetProperty .

- "Właściwości" na stronie 1789
- "Metody komunikatów Read i Write" na stronie 1795
- "Metody buforu" na stronie 1798
- "Metody właściwości" na stronie 1799
- "Konstruktory" na stronie 1801

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public string AccountingToken {get; set;}
Część kontekstu tożsamości komunikatu. Ułatwia to aplikacji naliczanie opłat za pracę wykonaną w wyniku komunikatu. Wartością domyślną jest MQC.MQACT_NONE.

public string ApplicationIdData {get; set;}
Część kontekstu tożsamości komunikatu. ApplicationIdDane to informacje definiowane przez pakiet aplikacji, które mogą służyć do udostępniania dodatkowych informacji o komunikacie lub jego twórcy. Wartością domyślną jest " ".

public string ApplicationOriginData {get; set;}
Informacje zdefiniowane przez aplikację, których można użyć do udostępnienia dodatkowych informacji o pochodzeniu komunikatu. Wartością domyślną jest " ".

public int BackoutCount {get;}
Liczba wskazująca, ile razy komunikat został poprzednio zwrócony i wycofany przez wywołanie MQQueue.Get w ramach jednostki pracy. Wartość domyślna to zero.

public int CharacterSet {get; set;}

Identyfikator kodowanego zestawu znaków danych znakowych w komunikacie.

Ustaw `CharacterSet` , aby identyfikować zestaw danych znakowych w komunikacie. Pobierz `CharacterSet` , aby dowiedzieć się, w jakim zestawie znaków użyto do kodowania danych znakowych w komunikacie.

Aplikacje .NET zawsze działają w kodowaniu Unicode, podczas gdy w innych środowiskach działają w tym samym zestawie znaków, w którym działa menedżer kolejek.

Metody `ReadString` i `ReadLine` dokonują konwersji danych znakowych w komunikacie do formatu Unicode.

Metoda `WriteString` przekształca kod Unicode w zestaw znaków zakodowany w `CharacterSet`. Jeśli właściwość `CharacterSet` jest ustawiona na wartość domyślną `MQC.MQCCSI_Q_MGR`, która wynosi 0, nie jest wykonywana konwersja, a `CharacterSet` jest ustawiona na 1200. Jeśli `CharacterSet` zostanie ustawiona na inną wartość, `WriteString` konwertuje kod Unicode na wartość alternatywną.

Uwaga: Inne metody odczytu i zapisu nie używają `CharacterSet`.

- `ReadChar` i `WriteChar` odczytują i zapisują znak Unicode do i z buforu komunikatów bez konwersji.
- `ReadUTF` i `WriteUTF` dokonują konwersji między łańcuchem Unicode w aplikacji a łańcuchem UTF-8 poprzedzonym polem o długości 2 bajtów w buforze komunikatów.
- Metody bajtowe przesyłają bez zmian bajty między aplikacją a buforem komunikatów.

public byte[] CorrelationId {get; set;}

- W przypadku wywołania `MQQueue.Get` jest to identyfikator korelacji komunikatu, który ma zostać pobrany. Menedżer kolejek zwraca pierwszy komunikat z identyfikatorem komunikatu i identyfikatorem korelacji zgodnym z polami deskryptora komunikatu. Wartość domyślna `MQC.MQCI_NONE`pomaga dopasować dowolny identyfikator korelacji.
- W przypadku wywołania `MQQueue.Put` jest to identyfikator korelacji, który ma zostać ustawiony.

public int DataLength {get;}
Liczba bajtów danych komunikatu, które pozostały do odczytania.

public int DataOffset {get; set;}
Bieżąca pozycja kursora w danych komunikatu. Odczyty i zapisy są uwzględniane w bieżącej pozycji.

public int Encoding {get; set;}
Reprezentacja używana dla wartości liczbowych w danych komunikatu aplikacji. Kodowanie ma zastosowanie do danych binarnych, dziesiętnych upakowanych i zmiennopozycyjnych. Zachowanie metod odczytu i zapisu dla tych formatów liczbowych zostało odpowiednio zmienione. Utwórz wartość dla pola kodowania, dodając jedną wartość z każdej z tych trzech sekcji. Alternatywnie można utworzyć wartość łączącą wartości z każdej z trzech sekcji za pomocą bitowego operatora OR.

1. Binarna liczba całkowita

   MQC.MQENC_INTEGER_NORMAL
   Liczby całkowite w układzie big endian.

   MQC.MQENC_INTEGER_REVERSED
   Liczby całkowite z układem little endian, używane w architekturze Intel .

2. Upakowane dziesiętne

   MQC.MQENC_DECIMAL_NORMAL
   Upakowana liczba dziesiętna z układem big endian, używana przez z/OS.

   MQC.MQENC_DECIMAL_REVERSED
   Little-endian packed-decimal.

3. zmiennopozycyjne

   MQC.MQENC_FLOAT_IEEE_NORMAL
   Zmiennopozycyjne IEEE Big-endian.

   MQC.MQENC_FLOAT_IEEE_REVERSED
   Zmiennopozycyjne IEEE w układzie little endian, zgodnie z używaną architekturą Intel .

MQC.MQENC_FLOAT_S390
z/OS format floating points (zmiennopozycyjne).

Wartość domyślna to:

```
MQC.MQENC_INTEGER_REVERSED |
MQC.MQENC_DECIMAL_REVERSED |
MQC.MQENC_FLOAT_IEEE_REVERSED
```

Ustawienie domyślne powoduje, że `WriteInt` zapisuje liczbę całkowitą w układzie little-endian, a `ReadInt` odczytuje liczbę całkowitą w układzie little-endian. Jeśli flaga `MQC.MQENC_INTEGER_NORMAL` zostanie ustawiona zamiast niej, program `WriteInt` zapisze liczbę całkowitą big endian, a program `ReadInt` odczyta liczbę całkowitą big endian.

Uwaga: Podczas konwersji liczb zmiennopozycyjnych w formacie IEEE na liczby zmiennopozycyjne w formacie zSeries może wystąpić utrata precyzji.

public int Expiry {get; set;}
Czas utraty ważności wyrażony w dziesiątych częściach sekundy, ustawiony przez aplikację, która umieszcza komunikat. Po upływie czasu utraty ważności komunikatu jest on zakwalifikowany do usunięcia przez menedżer kolejek. Jeśli w komunikacie określono jedną z flag `MQC.MQRO_EXPIRATION` , raport jest generowany po odrzucaniu komunikatu. Wartością domyślną jest `MQC.MQEI_UNLIMITED`, co oznacza, że komunikat nigdy nie traci ważności.

public int Feedback {get; set;}
Użyj opcji `Opinia` z komunikatem typu `MQC.MQMT_REPORT` , aby wskazać rodzaj raportu. System definiuje następujące kody sprzężenia zwrotnego:

- `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`

Można również użyć zdefiniowanych przez aplikację wartości sprzężenia zwrotnego z zakresu od `MQC.MQFB_APPL_FIRST` do `MQC.MQFB_APPL_LAST` . Wartością domyślną tego pola jest `MQC.MQFB_NONE`, co oznacza, że nie są udostępniane żadne informacje zwrotne.

public string Format {get; set;}
Nazwa formatu używana przez nadawcę komunikatu w celu wskazania odbiorcy rodzaju danych w komunikacie. Można używać własnych nazw formatów, ale nazwy rozpoczynające się od liter `MQ` mają znaczenie zdefiniowane przez menedżer kolejek. Wbudowane formaty menedżera kolejek są następujące:

MQC.MQFMT_ADMIN
Komunikat żądania/odpowiedzi serwera komend.

MQC.MQFMT_COMMAND_1
Wpisz 1 komunikat odpowiedzi na komendę.

MQC.MQFMT_COMMAND_2
Wpisz 2 komunikat odpowiedzi na komendę.

MQC.MQFMT_DEAD_LETTER_HEADER
Nagłówek niedostarczonego komunikatu.

MQC.MQFMT_EVENT
Komunikat zdarzenia.

MQC.MQFMT_NONE
Brak nazwy formatu.

MQC.MQFMT_PCF
Komunikat zdefiniowany przez użytkownika w formacie komend programowalnych.

MQC.MQFMT_STRING
Komunikat składający się w całości ze znaków.

MQC.MQFMT_TRIGGER
komunikat wyzwalacza

MQC.MQFMT_XMIT_Q_HEADER
Nagłówek kolejki transmisji.

Wartością domyślną jest `MQC.MQFMT_NONE`.

public byte[] GroupId {get; set;}
Łańcuch bajtów identyfikujący grupę komunikatów, do której należy komunikat fizyczny. Wartością domyślną jest `MQC.MQGI_NONE`.

public int MessageFlags {get; set;}
Flagi sterujące segmentacją i statusem komunikatu.

public byte[] MessageId {get; set;}

W przypadku wywołania `MQQueue.Get` pole to określa identyfikator komunikatu, który ma zostać pobrany. Zwykle menedżer kolejek zwraca pierwszy komunikat z identyfikatorem komunikatu i identyfikatorem korelacji, które są zgodne z polami deskryptora komunikatu. Zezwalaj na zgodność dowolnego identyfikatora komunikatu przy użyciu wartości specjalnej `MQC.MQMI_NONE`.

W przypadku wywołania `MQQueue.Put` pole to określa identyfikator komunikatu, który ma być używany. Jeśli określono `MQC.MQMI_NONE` , menedżer kolejek generuje unikalny identyfikator komunikatu podczas umieszczania komunikatu. Wartość tej zmiennej składowej jest aktualizowana po umieszczeniu, aby wskazać identyfikator komunikatu, który został użyty. Wartością domyślną jest `MQC.MQMI_NONE`.

public int MessageLength {get;}
Liczba bajtów danych komunikatu w obiekcie `MQMessage` .

public int MessageSequenceNumber {get; set;}
Numer kolejny komunikatu logicznego w grupie.

public int MessageType {get; set;}
Wskazuje typ komunikatu. Następujące wartości są obecnie zdefiniowane przez system:

- `MQC.MQMT_DATAGRAM`
- `MQC.MQMT_REPLY`
- `MQC.MQMT_REPORT`
- `MQC.MQMT_REQUEST`

Można również użyć wartości zdefiniowanych przez aplikację z zakresu od `MQC.MQMT_APPL_FIRST` do `MQC.MQMT_APPL_LAST`. Wartością domyślną tego pola jest `MQC.MQMT_DATAGRAM`.

public int Offset {get; set;}
W komunikacie segmentowanym: przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego.

public int OriginalLength {get; set;}
Oryginalna długość segmentowanego komunikatu.

public int Persistence {get; set;}

Trwałość komunikatu. Zdefiniowane są następujące wartości:

- MQC.MQPER_NOT_PERSISTENT

  Jeśli ta opcja zostanie ustawiona w kliencie z możliwością ponownego połączenia, kod przyczyny MQRC_NONE zostanie zwrócony do aplikacji po pomyślnym nawiązaniu połączenia.

- MQC.MQPER_PERSISTENT

  Jeśli ta opcja zostanie ustawiona w kliencie z możliwością ponownego nawiązania połączenia, kod przyczyny MQRC_CALL_INTERRUPTED zostanie zwrócony do aplikacji po pomyślnym nawiązaniu połączenia.

- MQC.MQPER_PERSISTENCE_AS_Q_DEF

Wartością domyślną jest MQC.MQPER_PERSISTENCE_AS_Q_DEF, która pobiera trwałość komunikatu z domyślnego atrybutu trwałości kolejki docelowej.

public int Priority {get; set;}
Priorytet komunikatu. Wartość specjalną MQC.MQPRI_PRIORITY_AS_Q_DEF można również ustawić w komunikacie wychodzącym. Priorytet komunikatu jest następnie pobierany z domyślnego atrybutu priorytetu kolejki docelowej. Wartością domyślną jest MQC.MQPRI_PRIORITY_AS_Q_DEF.

public int PropertyValidation {get; set;}
Określa, czy sprawdzanie poprawności właściwości ma miejsce po ustawieniu właściwości komunikatu. Dozwolone są następujące wartości:

- MQCMHO_DEFAULT_VALIDATION
- MQCMHO_VALIDATE
- MQCMHO_NO_VALIDATION

Wartością domyślną jest MQCMHO_DEFAULT_VALIDATION.

public string PutApplicationName {get; set;}
Nazwa aplikacji, która umieściła komunikat. Wartością domyślną jest "".

public int PutApplicationType {get; set;}
Typ aplikacji, która wstawiła komunikat. PutApplicationTyp może być wartością zdefiniowaną przez system lub wartością zdefiniowaną przez użytkownika. System definiuje następujące wartości:

- 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

Wartością domyślną jest MQC.MQAT_NO_CONTEXT, co oznacza, że w komunikacie nie ma informacji o kontekście.

public DateTime PutDateTime {get; set;}
Data i godzina umieszczenia komunikatu.

public string ReplyToQueueManagerName {get; set;}
Nazwa menedżera kolejek, który ma wysyłać komunikaty odpowiedzi lub raporty. Wartością domyślną jest "", a menedżer kolejek udostępnia właściwość ReplyToQueueManagerNazwa.

public string ReplyToQueueName {get; set;}
Nazwa kolejki komunikatów, do której aplikacja, która wysłała żądanie pobrania komunikatu, wysyła komunikaty `MQC.MQMT_REPLY` i `MQC.MQMT_REPORT` . Wartością domyślną parametru `ReplyToQueueName` jest `""`.

public int Report {get; set;}
Użyj opcji `Raport` , aby określić opcje dotyczące komunikatów raportu i odpowiedzi:

- Określa, czy raporty są wymagane.
- Określa, czy dane komunikatu aplikacji mają być uwzględniane w raportach.
- Sposób ustawiania identyfikatorów komunikatu i korelacji w raporcie lub w odpowiedzi.

Można zażądać dowolnej kombinacji czterech typów raportów:

- Określ dowolną kombinację czterech typów raportów. Wybór jednej z trzech opcji dla każdego typu raportu, w zależności od tego, czy dane komunikatu aplikacji mają być uwzględnione w komunikacie raportu.

  1. Potwierdź przy nadejściu
     - `MQC.MQRO_COA`
     - `MQC.MQRO_COA_WITH_DATA`
     - `MQC.MQRO_COA_WITH_FULL_DATA` **
  2. Potwierdź przy dostarczeniu
     - `MQC.MQRO_COD`
     - `MQC.MQRO_COD_WITH_DATA`
     - `MQC.MQRO_COD_WITH_FULL_DATA` **
  3. Wyjątek
     - `MQC.MQRO_EXCEPTION`
     - `MQC.MQRO_EXCEPTION_WITH_DATA`
     - `MQC.MQRO_EXCEPTION_WITH_FULL_DATA` **
  4. Termin ważności
     - `MQC.MQRO_EXPIRATION`
     - `MQC.MQRO_EXPIRATION_WITH_DATA`
     - `MQC.MQRO_EXPIRATION_WITH_FULL_DATA` **

  Uwaga: Wartości oznaczone na liście znacznikiem ** nie są obsługiwane przez menedżery kolejek produktu z/OS . Nie należy ich używać, jeśli aplikacja może uzyskać dostęp do menedżera kolejek produktu z/OS bez względu na platformę, na której aplikacja jest uruchomiona.
- Określ jedną z następujących opcji, aby kontrolować sposób generowania identyfikatora komunikatu dla raportu lub komunikatu odpowiedzi:
  - `MQC.MQRO_NEW_MSG_ID`
  - `MQC.MQRO_PASS_MSG_ID`
- Określ jedną z następujących opcji, aby sterować sposobem ustawiania identyfikatora korelacji komunikatu raportu lub odpowiedzi:
  - `MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID`
  - `MQC.MQRO_PASS_CORREL_ID`
- Określ jedną z następujących opcji, aby sterować umieszczaniem oryginalnego komunikatu, gdy nie może on zostać dostarczony do kolejki docelowej:
  - `MQC.MQRO_DEAD_LETTER_Q`
  - `MQC.MQRO_DISCARD_MSG` **
- Jeśli nie określono żadnych opcji raportu, wartością domyślną jest:

```
MQC.MQRO_NEW_MSG_ID |
MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID |
MQC.MQRO_DEAD_LETTER_Q
```

- Można określić jedną lub obie z poniższych opcji, aby zażądać, aby aplikacja odbierająca wysłała komunikat o działaniu pozytywnym lub negatywnym.
  - `MQC.MQRO_PAN`
  - `MQC.MQRO_NAN`

public int TotalMessageLength {get;}
Łączna liczba bajtów w komunikacie zapisanych w kolejce komunikatów, z której odebrano ten komunikat.

public string UserId {get; set;}
`UserId` jest częścią kontekstu tożsamości komunikatu. Wartość ta jest zwykle udostępniana przez menedżer kolejek. Wartość tę można nadpisać, jeśli użytkownik ma uprawnienia do ustawiania kontekstu tożsamości.

public int Version {get; set;}
Wersja używanej struktury MQMD.

Metody komunikatów Read i `Write`

Metody `Read` i `Write` wykonują te same funkcje, co składowe klas `BinaryReader` i `BinaryWriter` w przestrzeni nazw .NET `System.IO` . Pełna składnia języka i przykłady użycia znajdują się w sekcji MSDN. Metody odczytane lub zapisane z bieżącej pozycji w buforze komunikatów. Przenoszą one bieżącą pozycję do przodu o liczbę odczytanych lub zapisanych bajtów.

Uwaga: Jeśli dane komunikatu zawierają nagłówek MQRFH lub MQRFH2 , należy użyć metody `ReadBytes` , aby odczytać dane.

- Wszystkie metody zgłaszają wyjątek `IOException`.
- Metody `ReadFully` automatycznie dopasowują wielkość docelowej tablicy `byte` lub `sbyte` dokładnie do komunikatu. Wielkość tablicy o wartości NULL jest również zmieniana.
- `Read` metody throw `EndOfStreamException`.
- `WriteDecimal` metody throw `MQException`.
- Metody `ReadString`, `ReadLine` i `WriteString` przekształcają kod Unicode na zestaw znaków komunikatu; patrz `CharacterSet` .
- Metody `Decimal` odczytują i zapisują upakowane liczby dziesiętne zakodowane w formacie big endian, `MQC.MQENC_DECIMAL_NORMAL`lub little endian `MQC.MQENC_DECIMAL_REVERSE` zgodnie z wartością właściwości `Encoding`. Zakresy dziesiętne i odpowiadające im typy .NET są następujące:

  Decimal2/short
  Od -999 do 999

  Decimal4/int
  od -9999999 do 9999999

  Decimal8/long
  Od -999999999999999 do 999999999999999
- Metody `Double` i `Float` odczytują i zapisują wartości zmiennopozycyjne zakodowane w formatach IEE big-endian i little-endian, `MQC.MQENC_FLOAT_IEEE_NORMAL` i `MQC.MQENC_FLOAT_IEEE_REVERSED`lub w formacie S/390 , `MQC.MQENC_FLOAT_S390`, zgodnie z wartością parametru `Encoding`.
- Metody `Int` odczytują i zapisują wartości całkowite zakodowane w formacie big endian, `MQC.MQENC_INTEGER_NORMAL`lub little endian, `MQC.MQENC_INTEGER_REVERSED`, zgodnie z wartością właściwości `Encoding`. Wszystkie liczby całkowite są ze znakiem, z wyjątkiem dodania 2-bajtowego typu całkowitoliczbowego bez znaku. Wielkości całkowite oraz typy .NET i IBM MQ są następujące:

2 bajt
`short`, `Int2`, `ushort`, `UInt2`

4 bajty
`int`, `Int4`

8 bajtów
`long`, `Int8`

- Program `WriteObject` przesyła klasę obiektu, wartości jego pól innych niż przejściowe i niestatyczne oraz pola jego nadtypów do buforu komunikatów.
- `ReadObject` tworzy obiekt na podstawie klasy obiektu, sygnatury klasy oraz wartości jego pól nieprzejściowych i niestatycznych, a także pól ich nadtypów.

Tabela 843. Metody odczytu i zapisu komunikatów

Typ docelowy	Sygnatury metod

Boolean	`public bool ReadBoolean();`
	
	`public void WriteBoolean(bool value);`

Byte	`public byte ReadByte()`
	`public byte ReadUnsignedByte()`
	
	`public void Write(int value)`
	`public void WriteByte(int value)`
	`public void WriteByte(byte value)`
	`public void WriteByte(sbyte value)`

Bytes	`public byte[] ReadBytes(int count)`
	`public void ReadFully(ref byte[] value)`
	`public void ReadFully(ref sbyte[] value)`
	`public void ReadFully(ref byte[] value, int offset,int length)`
	`public void ReadFully(ref sbyte[] value, int offset,int length)`
	
	`public void Write(byte[] value)`
	`public void Write(sbyte[] value)`
	`public void Write(byte[] value, int offset,int length)`
	`public void Write(sbyte[] value, int offset,int length)`
	`public void WriteBytes(string value)`

Decimal2	`public void WriteDecimal2(short value)`
Decimal4	`public void WriteDecimal4(short value)`
Decimal8	`public void WriteDecimal8(short value)`

Tabela 843. Metody odczytu i zapisu komunikatów (kontynuacja)

Typ docelowy	Sygnatury metod
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)`

Tabela 843. Metody odczytu i zapisu komunikatów (kontynuacja)	
Typ docelowy	**Sygnatury metod**
string	`public string ReadString(int length)` `public void WriteString(string string)`
Unsigned Short	`public ushort ReadUnsignedShort()` `public ushort ReadUInt2()`

Unicode	`public string ReadLine()`
	`public char ReadChar()`
	`public void WriteChar(int value)`
	`public void WriteChars(string string)`

UTF	`public string ReadUTF()` `public void WriteUTF(string string)`

Metody buforu

public void ClearMessage();

Zwraca wyjątek IOException.

Usuwa wszystkie dane w buforze komunikatów i ustawia przesunięcie danych z powrotem na zero.

public void ResizeBuffer(int *size*)

Zwraca wyjątek IOException.

Wskazówka do obiektu MQMessage dotycząca wielkości buforu, która może być wymagana dla kolejnych operacji pobierania. Jeśli komunikat zawiera dane komunikatu, a nowa wielkość jest mniejsza niż bieżąca, dane komunikatu są obcinane.

public void Seek(int *pos*)

Zwraca wyjątek IOException, ArgumentOutOfRangeException, ArgumentException.

Przenosi kursor do pozycji bezwzględnej w buforze komunikatów określonej przez parametr *pos*. Kolejne odczyty i zapisy działają w tej pozycji w buforze.

public int SkipBytes(int *i*)

Zwraca wyjątek IOException, EndOfStreamException.

Przenosi do przodu n bajtów w buforze komunikatów i zwraca nliczbę pominiętych bajtów.

Metoda SkipBytes blokuje się do momentu wystąpienia jednego z następujących zdarzeń:

- Wszystkie bajty są pomijane
- Wykryto koniec buforu komunikatów
- Zgłoszono wyjątek

Metody właściwości

public void DeleteProperty(string *name*);

Zwraca wyjątek MQException.

Usuwa z komunikatu właściwość o podanej nazwie.

nazwa
Nazwa właściwości do usunięcia.

public System.Collections.IEnumerator GetPropertyNames(string *name*)

Zwraca wyjątek MQException.

Zwraca IEnumerator wszystkich nazw właściwości zgodnych z podaną nazwą. Znaku procentu '%' można użyć na końcu nazwy jako znaku wieloznacznego w celu filtrowania właściwości komunikatu, dopasowania do zera lub większej liczby znaków, w tym kropki.

nazwa
Nazwa właściwości do dopasowania.

Metody SetProperty i GetProperty
Wszystkie metody SetProperty i GetProperty zgłaszają wyjątek MQException.

Metoda SetProperty klasy MQMessage .NET dodaje nową właściwość, jeśli właściwość jeszcze nie istnieje. Jeśli jednak właściwość już istnieje, podana wartość właściwości jest dodawana na końcu listy. Gdy wiele wartości jest ustawianych na nazwę właściwości za pomocą funkcji SetProperty, wywołanie funkcji GetProperty dla tej nazwy powoduje zwrócenie tych wartości sekwencyjnie w kolejności, w jakiej zostały ustawione.

Zachowanie jest takie samo dla wszystkich metod o typie Set*Property i Get*Property , takich jak GetLongProperty, SetLongProperty, GetBooleanProperty, SetBooleanProperty, GetStringPropertyi SetStringProperty.

Tabela 844. Metody SetProperty i GetProperty

Typ	Sygnatury metod

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);`

Tabela 844. Metody `SetProperty` i `GetProperty` (kontynuacja)

Typ	Sygnatury metod

Double	`public double GetDoubleProperty(string name);`
	`public double GetDoubleProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetDoubleProperty(string name, double value);`
	`public void SetDoubleProperty(string name, MQPropertyDescriptor pd, double value);`

Float	`public float GetFloatProperty(string name);`
	`public float GetFloatProperty(string name, MQPropertyDescriptor pd);`
	
	`public void SetFloatProperty(string name, float value);`
	`public void SetFloatProperty(string name, MQPropertyDescriptor pd, float value);`

Int2	`public short GetInt2Property(string name);`
	`public short GetInt2Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt2Property(string name, short value);`
	`public void SetInt2Property(string name, MQPropertyDescriptor pd, short value);`

Int4	`public int GetInt4Property(string name);`
	`public int GetInt4Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt4Property(string name, int value);`
	`public void SetInt4Property(string name, MQPropertyDescriptor pd, int value);`

Int8	`public long GetInt8Property(string name);`
	`public long GetInt8Property(string name, MQPropertyDescriptor pd);`
	
	`public void SetInt8Property(string name, long value);`
	`public void SetInt8Property(string name, MQPropertyDescriptor pd, long value);`

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);`

Tabela 844. Metody SetProperty i GetProperty (kontynuacja)

Typ	Sygnatury metod

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);`

Konstruktory

public MQMessage();

Tworzy obiekt MQMessage z domyślną informacją deskryptora komunikatu i pustym buforem komunikatu.

Klasa MQProcess.NET

Użyj MQProcess , aby wysłać zapytanie o atrybuty procesu IBM MQ . Utwórz obiekt MQProcess przy użyciu konstruktora lub metody MQQueueManager AccessProcess .

Klasa

```
System.Object
        |
        └─ IBM.WMQ.MQBase
                |
                └─ IBM.WMQ.MQBaseObject
                        |
                        └─ IBM.WMQ.MQManagedObject
                                |
                                └─ IBM.WMQ.MQProcess
```

```
public class IBM.WMQ.MQProcess extends IBM.WMQ.MQManagedObject;
```

- “Właściwości” na stronie 1801
- “Konstruktory” na stronie 1802

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public string ApplicationId {get;}

Pobiera łańcuch znaków identyfikujący aplikację, która ma zostać uruchomiona. ApplicationId jest używany przez aplikację monitora wyzwalacza. ApplicationId jest wysyłany do kolejki inicjującej jako część komunikatu wyzwalacza.

Wartością domyślną jest NULL.

public int ApplicationType {get;}

Określa typ procesu, który ma zostać uruchomiony przez aplikację monitora wyzwalacza. Zdefiniowane są typy standardowe, ale można użyć innych typów:

- MQAT_AIX
- MQAT_CICS
- MQAT_IMS
- MQAT_MVS
- MQAT_NATIVE
- MQAT_OS400
- MQAT_UNIX
- MQAT_WINDOWS
- MQAT_JAVA
- MQAT_USER_FIRST
- MQAT_USER_LAST

Wartością domyślną jest MQAT_NATIVE.

public string EnvironmentData {get;}

Pobiera informacje o środowisku aplikacji, która ma zostać uruchomiona.

Wartością domyślną jest NULL.

public string UserData {get;}

Pobiera informacje podane przez użytkownika o aplikacji, która ma zostać uruchomiona.

Wartością domyślną jest NULL.

Konstruktory

public MQProcess(MQQueueManager *queueManager*, string *processName*, int *openOptions*);
public MQProcess(MQQueueManager *qMgr*, string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Zwraca wyjątek MQException.

Uzyskaj dostęp do procesu IBM MQ w menedżerze kolejek *qMgr* , aby uzyskać dostęp do atrybutów procesu.

qMgr
Menedżer kolejek, do którego ma zostać uzyskany dostęp.

processName
Nazwa procesu, który ma zostać otwarty.

openOptions
Opcje sterujące otwieraniem procesu. Poprawne opcje, które można dodać lub połączyć za pomocą bitowej operatora OR, to:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

QueueManagerName
Nazwa menedżera kolejek, w którym jest zdefiniowany proces. Jeśli menedżer kolejek jest taki sam, jak menedżer, do którego proces uzyskuje dostęp, można pozostawić pustą lub pustą nazwę menedżera kolejek.

Identyfikator alternateUser

Jeśli parametr **openOptions** ma wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY , parametr *alternateUserId* określa alternatywny ID użytkownika używany do sprawdzania autoryzacji dla działania. Jeśli parametr MQOO_ALTERNATE_USER_AUTHORITY nie jest określony, parametr *alternateUserId* może być pusty lub mieć wartość null.

Jeśli nie określono opcji MQC.MQOO_ALTERNATE_USER_AUTHORITY , do połączenia z menedżerem kolejek używane jest domyślne uprawnienie użytkownika.

public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*);
public MQProcess MQQueueManager.AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Zwraca wyjątek MQException.

Uzyskaj dostęp do procesu IBM MQ w tym menedżerze kolejek, aby uzyskać informacje o atrybutach procesu.

processName
Nazwa procesu, który ma zostać otwarty.

openOptions
Opcje sterujące otwieraniem procesu. Poprawne opcje, które można dodać lub połączyć za pomocą bitowej operatora OR, to:

- MQC.MQOO_FAIL_IF_QUIESCING
- MQC.MQOO_INQUIRE
- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

QueueManagerName
Nazwa menedżera kolejek, w którym jest zdefiniowany proces. Jeśli menedżer kolejek jest taki sam, jak menedżer, do którego proces uzyskuje dostęp, można pozostawić pustą lub pustą nazwę menedżera kolejek.

Identyfikator alternateUser

Jeśli parametr **openOptions** ma wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY , parametr *alternateUserId* określa alternatywny ID użytkownika używany do sprawdzania autoryzacji dla działania. Jeśli parametr MQOO_ALTERNATE_USER_AUTHORITY nie jest określony, parametr *alternateUserId* może być pusty lub mieć wartość null.

Jeśli nie określono opcji MQC.MQOO_ALTERNATE_USER_AUTHORITY , do połączenia z menedżerem kolejek używane jest domyślne uprawnienie użytkownika.

Klasa MQPropertyDescriptor.NET

Jako parametru metod MQMessage GetProperty i SetProperty należy użyć parametru MQPropertyDescriptor . MQPropertyDescriptor opisuje właściwość MQMessage .

Klasa

```
System.Object
        |
        └─ IBM.WMQ.MQPropertyDescriptor
```

```
public class IBM.WMQ.MQPropertyDescriptor extends System.Object;
```


- "Właściwości" na stronie 1804
- "Konstruktory" na stronie 1805

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public int Context {get; set;}

Kontekst komunikatu, do którego należy właściwość. Dozwolone są następujące wartości:

MQC.MQPD_NO_CONTEXT
Właściwość nie jest powiązana z kontekstem komunikatu.

MQC.MQPD_USER_CONTEXT
Właściwość jest powiązana z kontekstem użytkownika.

Jeśli użytkownik jest autoryzowany, właściwość powiązana z kontekstem użytkownika jest zapisywana podczas pobierania komunikatu. Kolejna metoda Put odwołująca się do zapisanego kontekstu może przekazać właściwość do nowego komunikatu.

public int CopyOptions {get; set;}

CopyOptions opisuje typ komunikatu, do którego można skopiować właściwość.

Gdy menedżer kolejek odbiera komunikat zawierający właściwość zdefiniowaną przez IBM MQ , którą menedżer kolejek rozpoznaje jako niepoprawną, poprawia wartość pola CopyOptions .

Można podać dowolną kombinację poniższych opcji. Połącz opcje, dodając wartości lub używając bitowej metody OR.

MQC.MQCOPY_ALL
Właściwość jest kopiowana do wszystkich typów kolejnych komunikatów.

MQC.MQCOPY_FORWARD
Właściwość jest kopiowana do przekazywanego komunikatu.

MQC.MQCOPY_PUBLISH
Właściwość jest kopiowana do komunikatu odebranego przez subskrybent podczas publikowania komunikatu.

MQC.MQCOPY_REPLY
Właściwość jest kopiowana do komunikatu odpowiedzi.

MQC.MQCOPY_REPORT
Właściwość jest kopiowana do komunikatu raportu.

MQC.MQCOPY_DEFAULT
Wartość wskazuje, że nie określono innych opcji kopiowania. Nie istnieje relacja między właściwością a kolejnymi komunikatami. Wartość MQC.MQCOPY_DEFAULT jest zawsze zwracana dla właściwości deskryptora komunikatu.

MQC.MQCOPY_NONE
Taki sam jak MQC.MQCOPY_DEFAULT

public int Options { set; }
Opcje domyślnie: CMQC.MQPD_NONE. Nie można ustawić żadnej innej wartości.

public int Support { get; set; }

Ustaw opcję Obsługa , aby określić poziom obsługi wymagany dla właściwości komunikatów zdefiniowanych przez IBM MQ. Obsługa wszystkich innych właściwości jest opcjonalna. Można podać dowolną z następujących wartości lub nie można podać żadnej z nich

MQC.MQPD_SUPPORT_OPTIONAL
Właściwość jest akceptowana przez menedżer kolejek, nawet jeśli nie jest obsługiwana. Właściwość można odrzucić, aby komunikat przepływał do menedżera kolejek, który nie obsługuje właściwości komunikatu. Ta wartość jest również przypisywana do właściwości, które nie są zdefiniowane w systemie IBM MQ .

MQC.MQPD_SUPPORT_REQUIRED
Obsługa tej właściwości jest wymagana. Jeśli komunikat zostanie umieszczony w menedżerze kolejek, który nie obsługuje właściwości zdefiniowanej przez IBM MQ,

metoda nie powiedzie się. Zwraca kod zakończenia MQC.MQCC_FAILED i kod przyczyny MQC.MQRC_UNSUPPORTED_PROPERTY.

MQC.MQPD_SUPPORT_REQUIRED_IF_LOCAL

Obsługa tej właściwości jest wymagana, jeśli komunikat jest przeznaczony dla kolejki lokalnej. Jeśli komunikat zostanie umieszczony w kolejce lokalnej w menedżerze kolejek, który nie obsługuje właściwości zdefiniowanej przez IBM MQ, wykonanie metody nie powiedzie się. Zwraca kod zakończenia MQC.MQCC_FAILED i kod przyczyny MQC.MQRC_UNSUPPORTED_PROPERTY.

Jeśli komunikat jest umieszczany w zdalnym menedżerze kolejek, nie jest wykonywane żadne sprawdzenie.

Konstruktory

PropertyDescriptor();
Utwórz deskryptor właściwości.

Klasa MQPutMessageOptions.NET

Parametr MQPutMessageOptions umożliwia określenie sposobu wysyłania komunikatów. Modyfikuje on zachowanie MQDestination.Put.

Klasa

```
System.Object
        |
        └─ IBM.WMQ.MQBase
                |
                └─ IBM.WMQ.MQBaseObject
                        |
                        └─ IBM.WMQ.MQPutMessageOptions
```

```
public class IBM.WMQ.MQPutMessageOptions extends IBM.WMQ.MQBaseObject;
```

- "Właściwości" na stronie 1805 "Konstruktory" na stronie 1808

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

Uwaga: Zachowanie niektórych opcji dostępnych w tej klasie zależy od środowiska, w którym są używane. Te elementy są oznaczone gwiazdką (*).

public MQQueue ContextReference {get; set;}

Jeśli pole options zawiera łańcuch MQC.MQPMO_PASS_IDENTITY_CONTEXT lub MQC.MQPMO_PASS_ALL_CONTEXT, należy ustawić to pole tak, aby odwoływał się do pliku MQQueue , z którego mają zostać pobrane informacje o kontekście.

Wartość początkowa tego pola jest pusta.

public int InvalidDestCount {get;}*

Zwykle używany dla list dystrybucyjnych, InvalidDestLiczba wskazuje liczbę komunikatów, które nie mogły zostać wysłane do kolejek na liście dystrybucyjnej. Liczba ta obejmuje kolejki, których otwarcie nie powiodło się, oraz kolejki, które zostały pomyślnie otwarte, ale dla których operacja umieszczania nie powiodła się.

.NET nie obsługuje list dystrybucyjnych, ale podczas otwierania pojedynczej kolejki ustawiana jest wartość InvalidDestCount .

public int KnownDestCount {get;} *

Zwykle używany dla list dystrybucyjnych, KnownDestCount wskazuje liczbę komunikatów, które zostały pomyślnie wysłane przez bieżące wywołanie do kolejek, które zostały przetłumaczone na kolejki lokalne.

.NET nie obsługuje list dystrybucyjnych, ale podczas otwierania pojedynczej kolejki ustawiana jest wartość InvalidDestCount .

public int Options {get; set;}

Opcje, które sterują działaniem MQDestination.put i MQQueueManager.put. Można podać dowolną z poniższych wartości lub nie podać żadnej z nich. Jeśli wymagana jest więcej niż jedna opcja, wartości mogą być dodawane lub łączone za pomocą operatora OR.

MQC.MQPMO_ASYNC_RESPONSE
Ta opcja powoduje, że wywołanie MQDestination.put jest wykonywane asynchronicznie, z pewnymi danymi odpowiedzi.

MQC.MQPMO_DEFAULT_CONTEXT
Powiąż kontekst domyślny z komunikatem.

MQC.MQPMO_FAIL_IF_QUIESCING
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

MQC.MQPMO_LOGICAL_ORDER *
Umieść logiczne komunikaty i segmenty w grupach komunikatów w kolejności logicznej.

Jeśli opcja MQPMO_LOGICAL_ORDER zostanie użyta w kliencie z możliwością ponownego połączenia, kod przyczyny MQRC_RECONNECT_INCOMPATIBLE zostanie zwrócony do aplikacji.

MQC.MQPMO_NEW_CORREL_ID *
Wygeneruj nowy identyfikator korelacji dla każdego wysłanego komunikatu.

MQC.MQPMO_NEW_MSG_ID *
Wygeneruj nowy identyfikator dla każdego wysłanego komunikatu.

MQC.MQPMO_NONE
Nie określono żadnych opcji. Nie należy używać z innymi opcjami.

MQC.MQPMO_NO_CONTEXT
Z komunikatem nie jest powiązany żaden kontekst.

MQC.MQPMO_NO_SYNCPOINT
Umieść komunikat bez elementu sterującego punktu synchronizacji. Jeśli opcja kontroli punktu synchronizacji nie jest określona, przyjmowana jest wartość domyślna bez punktu synchronizacji.

MQC.MQPMO_PASS_ALL_CONTEXT
Przekaż cały kontekst z uchwytu kolejki wejściowej.

MQC.MQPMO_PASS_IDENTITY_CONTEXT
Przekaż kontekst tożsamości z uchwytu kolejki wejściowej.

MQC.MQPMO_RESPONSE_AS_Q_DEF

W przypadku wywołania MQDestination.put ta opcja pobiera typ odpowiedzi put z atrybutu DEFPRESP kolejki.

W przypadku wywołania MQQueueManager.put ta opcja powoduje, że wywołanie jest wykonywane synchronicznie.

MQC.MQPMO_RESPONSE_AS_TOPIC_DEF
MQC.MQPMO_RESPONSE_AS_TOPIC_DEF to synonim terminu MQC.MQPMO_RESPONSE_AS_Q_DEF do użycia z obiektami tematów.

MQC.MQPMO_RETAIN

Wysyłana publikacja ma zostać zachowana przez menedżer kolejek. Jeśli ta opcja jest używana i nie można zachować publikacji, komunikat nie jest publikowany i wywołanie kończy się niepowodzeniem z komunikatem MQC.MQRC_PUT_NOT_RETAINED.

Zażądaj kopii tej publikacji po jej opublikowaniu, wywołując metodę `MQSubscription.RequestPublicationUpdate` . Zapisana publikacja jest wysyłana do aplikacji, które tworzą subskrypcję bez ustawiania opcji `MQC.MQSO_NEW_PUBLICATIONS_ONLY` . Sprawdź właściwość komunikatu `MQIsRetained` publikacji po jej odebraniu, aby dowiedzieć się, czy była to zachowana publikacja.

Jeśli subskrybent żąda zachowanych publikacji, używana subskrypcja może zawierać znak wieloznaczny w łańcuchu tematu. Jeśli w drzewie tematów znajduje się wiele zachowanych publikacji zgodnych z subskrypcją, zostaną one wysłane.

MQC.MQPMO_SET_ALL_CONTEXT
Ustaw cały kontekst z aplikacji.

MQC.MQPMO_SET_IDENTITY_CONTEXT
Ustaw kontekst tożsamości z aplikacji.

MQC.MQPMO_SYNC_RESPONSE
Ta opcja powoduje, że wywołanie metody `MQDestination.put` lub `MQQueueManager.put` jest wykonywane synchronicznie z pełnymi danymi odpowiedzi.

MQC.MQPMO_SUPPRESS_REPLYTO
Wszystkie informacje wprowadzone w polach `ReplyToQueueName` i `ReplyToQueueManagerName` publikacji nie są przekazywane do subskrybentów. Jeśli ta opcja zostanie użyta w połączeniu z opcją raportu, która wymaga odpowiedzi `ReplyToQueueName`, wywołanie nie powiedzie się i zostanie wyświetlony komunikat `MQC.MQRC_MISSING_REPLY_TO_Q`.

MQC.MQPMO_SYNCPOINT
Umieść komunikat z elementem sterującym punktu synchronizacji. Komunikat nie jest widoczny poza jednostką pracy, dopóki jednostka pracy nie zostanie zatwierdzona. Jeśli jednostka pracy zostanie wycofana, komunikat zostanie usunięty.

public int RecordFields {get; set;} *

Informacje o listach dystrybucyjnych. Listy dystrybucyjne nie są obsługiwane w systemie .NET.

public string ResolvedQueueManagerName {get;}

Pole wyjściowe ustawione przez menedżer kolejek na nazwę menedżera kolejek, do którego należy kolejka określona przez nazwę kolejki zdalnej. `ResolvedQueueManagerName` może być inna niż nazwa menedżera kolejek, z którego uzyskano dostęp do kolejki, jeśli kolejka jest kolejką zdalną.

Wartość niepusta jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką. Jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

public string ResolvedQueueName {get;}

Pole wyjściowe ustawione przez menedżera kolejek na nazwę kolejki, w której umieszczony jest komunikat. `ResolvedQueueNazwa` może różnić się od nazwy użytej do otwarcia kolejki, jeśli otwarta kolejka była kolejką aliasową lub kolejką modelową.

Niepusta wartość jest zwracana tylko wtedy, gdy obiekt jest pojedynczą kolejką. Jeśli obiekt jest listą dystrybucyjną lub tematem, zwracana wartość jest niezdefiniowana.

public int UnknownDestCount {get;} *

Zwykle używane w przypadku list dystrybucyjnych `UnknownDestCount` jest polem wyjściowym ustawionym przez menedżera kolejek. Raportuje on liczbę komunikatów, które zostały pomyślnie wysłane przez bieżące wywołanie do kolejek, które zostały przetłumaczone na kolejki zdalne.

.NET nie obsługuje list dystrybucyjnych, ale podczas otwierania pojedynczej kolejki ustawiana jest wartość `InvalidDestCount` .

Konstruktory

public MQPutMessageOptions();

Utwórz nowy obiekt MQPutMessageOptions bez ustawionych opcji oraz puste ResolvedQueueNazwa i ResolvedQueueManagerName.

Klasa MQQueue.NET

Parametr MQQueue umożliwia wysyłanie i odbieranie komunikatów oraz wysyłanie zapytań o atrybuty kolejki produktu IBM MQ . Utwórz obiekt MQQueue przy użyciu konstruktora lub metody MQQueueManager.AccessProcess .

Klasa

```
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;
```


- "Właściwości" na stronie 1808
- "metody" na stronie 1810
- "Konstruktory" na stronie 1813

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public int ClusterWorkLoadPriority {get;}
Określa priorytet kolejki. Ten parametr jest poprawny tylko dla kolejek lokalnych, zdalnych i aliasowych.

public int ClusterWorkLoadRank {get;}
Określa klasyfikację kolejki. Ten parametr jest poprawny tylko dla kolejek lokalnych, zdalnych i aliasowych.

public int ClusterWorkLoadUseQ {get;}
Określa zachowanie operacji MQPUT, gdy kolejka docelowa ma instancję lokalną i co najmniej jedną instancję klastra zdalnego. Ten parametr nie ma zastosowania, jeśli MQPUT pochodzi z kanału klastra. Ten parametr jest poprawny tylko dla kolejek lokalnych.

public DateTime CreationDateTime {get;}
Data i godzina utworzenia tej kolejki.

public int CurrentDepth {get;}
Pobiera liczbę komunikatów znajdujących się obecnie w kolejce. Ta wartość jest zwiększana podczas wywołania put i backout wywołania get. Jest on zmniejszany podczas pobierania bez przeglądania i podczas wycofywania wywołania put.

public int DefinitionType {get;}
Sposób zdefiniowania kolejki. Możliwe wartości:

- MQC.MQQDT_PREDEFINED
- MQC.MQQDT_PERMANENT_DYNAMIC

- MQC.MQQDT_TEMPORARY_DYNAMIC

public int InhibitGet {get; set;}
Określa, czy komunikaty mogą być wyświetlane w tej kolejce, czy w tym temacie. Możliwe wartości:

- MQC.MQQA_GET_INHIBITED
- MQC.MQQA_GET_ALLOWED

public int InhibitPut {get; set;}
Określa, czy można umieszczać komunikaty w tej kolejce, czy w tym temacie. Możliwe wartości:

- MQQA_PUT_INHIBITED
- MQQA_PUT_ALLOWED

public int MaximumDepth {get;}
Maksymalna liczba komunikatów, które mogą istnieć w kolejce w dowolnym momencie. Próba umieszczenia komunikatu w kolejce, która już zawiera tę liczbę komunikatów, kończy się niepowodzeniem z kodem przyczyny MQC.MQRC_Q_FULL.

public int MaximumMessageLength {get;}
Maksymalna długość danych aplikacji, które mogą istnieć w każdym komunikacie w tej kolejce. Próba umieszczenia komunikatu większego niż ta wartość nie powiodła się z kodem przyczyny MQC.MQRC_MSG_TOO_BIG_FOR_Q.

public int NonPersistentMessageClass {get;}
Poziom niezawodności komunikatów nietrwałych umieszczanych w tej kolejce.

public int OpenInputCount {get;}
Liczba uchwytów, które są obecnie poprawne dla usuwania komunikatów z kolejki. OpenInputOpenInput to łączna liczba poprawnych uchwytów wejściowych znanych lokalnemu menedżerowi kolejek, a nie tylko uchwytów utworzonych przez aplikację.

public int OpenOutputCount {get;}
Liczba uchwytów, które są obecnie poprawne dla dodawania komunikatów do kolejki. OpenOutputOpenOutput to łączna liczba poprawnych uchwytów wyjścia znanych lokalnemu menedżerowi kolejek, a nie tylko uchwytów utworzonych przez aplikację.

public int QueueAccounting {get;}
Określa, czy można włączyć gromadzenie informacji rozliczeniowych dla kolejki.

public int QueueMonitoring {get;}
Określa, czy można włączyć monitorowanie kolejki.

public int QueueStatistics {get;}
Określa, czy można włączyć gromadzenie statystyk dla kolejki.

public int QueueType {get;}
Typ tej kolejki z jedną z następujących wartości:

- MQC.MQQT_ALIAS
- MQC.MQQT_LOCAL
- MQC.MQQT_REMOTE
- MQC.MQQT_CLUSTER

public int Shareability {get;}
Określa, czy kolejka może być wielokrotnie otwierana do wprowadzania. Możliwe wartości:

- MQC.MQQA_SHAREABLE
- MQC.MQQA_NOT_SHAREABLE

public string TPIPE {get;}
Nazwa TPIPE używana do komunikacji z OTMA za pomocą mostu IBM MQ IMS .

public int TriggerControl {get; set;}
Określa, czy komunikaty wyzwalacza są zapisywane w kolejce inicjującej w celu uruchomienia aplikacji obsługującej kolejkę. Możliwe wartości:

- MQC.MQTC_OFF
- MQC.MQTC_ON

public string TriggerData {get; set;}
Dane w formacie swobodnym, które menedżer kolejek wstawia do komunikatu wyzwalacza. Wstawia TriggerData , gdy komunikat docierający do tej kolejki powoduje zapisanie komunikatu wyzwalacza w kolejce inicjującej. Maksymalna dozwolona długość łańcucha jest określona przez parametr MQC.MQ_TRIGGER_DATA_LENGTH.

public int TriggerDepth {get; set;}
Liczba komunikatów, które muszą znajdować się w kolejce przed zapisaniem komunikatu wyzwalacza, gdy typ wyzwalacza jest ustawiony na MQC.MQTT_DEPTH.

public int TriggerMessagePriority {get; set;}
Priorytet komunikatu, przy którym komunikaty nie przyczyniają się do generowania komunikatów wyzwalacza. Oznacza to, że menedżer kolejek ignoruje te komunikaty podczas podejmowania decyzji o generowaniu wyzwalacza. Wartość zero powoduje, że wszystkie komunikaty przyczyniają się do generowania komunikatów wyzwalacza.

public int TriggerType {get; set;}
Warunki, w których komunikaty wyzwalacza są zapisywane w wyniku komunikatów przychodzących do tej kolejki. Możliwe wartości:

- MQC.MQTT_NONE
- MQC.MQTT_FIRST
- MQC.MQTT_EVERY
- MQC.MQTT_DEPTH

metody

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);

Zwraca wyjątek MQException.

Pobiera komunikat z kolejki.

Jeśli operacja pobierania nie powiedzie się, obiekt MQMessage pozostanie niezmieniony. Jeśli operacja powiedzie się, części deskryptora komunikatu i danych komunikatu w pliku MQMessage są zastępowane deskryptorem komunikatu i danymi komunikatu z komunikatu przychodzącego.

Wszystkie wywołania IBM MQ z określonego MQQueueManager są synchroniczne. Z tego powodu, jeśli wykonywana jest operacja get with wait, wszystkie inne wątki korzystające z tego samego MQQueueManager będą miały zablokowaną możliwość wykonywania kolejnych wywołań IBM MQ do czasu wykonania wywołania Get. Jeśli dostęp do programu IBM MQ ma być uzyskiwany jednocześnie z wielu wątków, każdy wątek musi utworzyć własny obiekt MQQueueManager .

komunikat
Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskryptorze komunikatu są parametrami wejściowymi. Należy upewnić się, że parametry wejściowe MessageId i CorrelationId są ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia dla komunikatów odebranych w ramach elementu MQGM_SYNCPOINT.

Opcje getMessage
Opcje sterujące działaniem pobierania.

Użycie opcji MQC.MQGMO_CONVERT może spowodować wystąpienie wyjątku z kodem przyczyny MQC.MQRC_CONVERTED_STRING_TOO_BIG podczas przekształcania kodów znaków

jednobajtowych w kody dwubajtowe. W takim przypadku komunikat jest kopiowany do buforu bez konwersji.

Jeśli parametr *getMessageOptions* nie jest określony, używana jest opcja komunikatu MQGMO_NOWAIT.

Jeśli opcja MQGMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

MaxMsg
Największy komunikat, który ma zostać odebrany przez ten obiekt komunikatu. Jeśli komunikat w kolejce jest większy niż ta wielkość, wystąpi jedna z dwóch sytuacji:

- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG jest ustawiona w obiekcie MQGetMessageOptions , komunikat jest wypełniany możliwie największą ilością danych komunikatu. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_ACCEPTED .
- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie jest ustawiona, komunikat pozostaje w kolejce. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_FAILED .

Jeśli parametr *MaxMsgSize* nie jest określony, pobierany jest cały komunikat.

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Zwraca wyjątek MQException.

Umieszcza komunikat w kolejce.

Modyfikacje obiektu MQMessage dokonane po wywołaniu metody Put nie mają wpływu na rzeczywisty komunikat w kolejce produktu IBM MQ lub w temacie publikacji.

Put aktualizuje właściwości MessageId i CorrelationId obiektu MQMessage i nie czyści danych komunikatu. Dalsze wywołania Put lub Get odnoszą się do zaktualizowanych informacji w obiekcie MQMessage . Na przykład w poniższym fragmencie kodu pierwszy komunikat zawiera a , a drugi ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

komunikat
Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat do wysłania. W wyniku tej metody można zmienić deskryptor komunikatu. Wartości w deskryptorze komunikatu bezpośrednio po zakończeniu tej metody są wartościami, które zostały umieszczone w kolejce lub opublikowane w temacie.

Do klienta z możliwością ponownego połączenia zwracane są następujące kody przyczyny:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało zerwane podczas wykonywania wywołania metody Put dla trwałego komunikatu i ponowne połączenie zakończyło się pomyślnie.
- MQRC_NONE , jeśli połączenie zostało pomyślnie nawiązane podczas wykonywania wywołania metody Put dla komunikatu nietrwałego (patrz sekcja Odtwarzanie aplikacji).

Opcje komendy putMessage
Opcje sterujące działaniem operacji put.

Jeśli parametr *putMessageOptions* nie zostanie podany, zostanie użyta domyślna instancja MQPutMessageOptons .

Jeśli opcja MQPMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

Uwaga: W celu uproszczenia i zwiększenia wydajności, jeśli pojedynczy komunikat ma zostać umieszczony w kolejce, należy użyć obiektu MQQueueManager.Put . W tym celu powinien istnieć obiekt MQQueue .

public void PutForwardMessage(MQMessage *message*);
public void PutForwardMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);

Zwraca wyjątek MQException

Umieść komunikat przekazywany do kolejki, gdzie *message* jest oryginalnym komunikatem.

komunikat
Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat do wysłania. W wyniku tej metody można zmienić deskryptor komunikatu. Wartości w deskryptorze komunikatu bezpośrednio po zakończeniu tej metody są wartościami, które zostały umieszczone w kolejce lub opublikowane w temacie.

Do klienta z możliwością ponownego połączenia zwracane są następujące kody przyczyny:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało zerwane podczas wykonywania wywołania metody Put dla trwałego komunikatu i ponowne połączenie zakończyło się pomyślnie.
- MQRC_NONE , jeśli połączenie zostało pomyślnie nawiązane podczas wykonywania wywołania metody Put dla komunikatu nietrwałego (patrz sekcja Odtwarzanie aplikacji).

Opcje komendy putMessage
Opcje sterujące działaniem operacji put.

Jeśli parametr *putMessageOptions* nie zostanie podany, zostanie użyta domyślna instancja MQPutMessageOptons .

Jeśli opcja MQPMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

public void PutReplyMessage(MQMessage *message*)
public void PutReplyMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Zwraca wyjątek MQException.

Umieść komunikat odpowiedzi w kolejce, gdzie *message* jest oryginalnym komunikatem.

komunikat
Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskryptorze komunikatu są parametrami wejściowymi. Należy upewnić się, że parametry wejściowe MessageId i CorrelationId są ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia dla komunikatów odebranych w ramach elementu MQGM_SYNCPOINT.

Opcje komendy putMessage
Opcje sterujące działaniem operacji put.

Jeśli parametr *putMessageOptions* nie zostanie podany, zostanie użyta domyślna instancja MQPutMessageOptons .

Jeśli opcja MQPMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

public void PutReportMessage(MQMessage *message*)
public void PutReportMessage(MQMessage *message*, MQPutMessageOptions *putMessageOptions*)

Zwraca wyjątek MQException.

Umieść komunikat raportu w kolejce, gdzie *message* jest oryginalnym komunikatem.

komunikat
Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskryptorze komunikatu są parametrami wejściowymi. Należy upewnić się, że parametry wejściowe `MessageId` i `CorrelationId` są ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny `MQRC_BACKED_OUT` po pomyślnym ponownym nawiązaniu połączenia dla komunikatów odebranych w ramach elementu `MQGM_SYNCPOINT`.

Opcje komendy putMessage
Opcje sterujące działaniem operacji put.

Jeśli parametr *`putMessageOptions`* nie zostanie podany, zostanie użyta domyślna instancja `MQPutMessageOptons` .

Jeśli opcja `MQPMO_LOGICAL_ORDER` jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny `MQRC_RECONNECT_INCOMPATIBLE` .

Konstruktory

`public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions);`
`public MQQueue MQQueueManager.AccessQueue(string queueName, int openOptions,`
`string queueManagerName, string dynamicQueueName, string alternateUserId);`

Zwraca wyjątek `MQException`.

Uzyskuje dostęp do kolejki w tym menedżerze kolejek.

Można pobrać lub przeglądać komunikaty, umieścić komunikaty, uzyskać informacje o atrybutach kolejki lub ustawić atrybuty kolejki. Jeśli kolejka o nazwie jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Wykonaj zapytanie o atrybut `name` wynikowego obiektu `MQQueue` , aby znaleźć nazwę kolejki dynamicznej.

queueName
Nazwa kolejki do otwarcia.

openOptions
Opcje sterujące otwieraniem kolejki.

`MQC.MQOO_ALTERNATE_USER_AUTHORITY`
Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

`MQC.MQOO_BIND_AS_QDEF`
Użyj domyślnego powiązania dla kolejki.

`MQC.MQOO_BIND_NOT_FIXED`
Nie wiąż się z konkretnym miejscem docelowym.

`MQC.MQOO_BIND_ON_OPEN`
Powiąż uchwyt z miejscem docelowym, gdy kolejka jest otwarta.

`MQC.MQOO_BROWSE`
Otwórz, aby przeglądać komunikat.

`MQC.MQOO_FAIL_IF_QUIESCING`
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

`MQC.MQOO_INPUT_AS_Q_DEF`
Otwórz, aby pobrać komunikaty przy użyciu wartości domyślnej zdefiniowanej w kolejce.

`MQC.MQOO_INPUT_SHARED`
Otwórz, aby uzyskać wiadomości z dostępem współużytkowanym.

`MQC.MQOO_INPUT_EXCLUSIVE`
Otwórz, aby uzyskać wiadomości z wyłącznym dostępem.

`MQC.MQOO_INQUIRE`
Otwórz dla zapytania-wymagane, jeśli chcesz wykonać zapytanie o właściwości.

MQC.MQOO_OUTPUT
Otwórz, aby umieścić komunikaty.

MQC.MQOO_PASS_ALL_CONTEXT
Zezwalaj na przekazywanie całego kontekstu.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Zezwalaj na przekazywanie kontekstu tożsamości.

MQC.MQOO_SAVE_ALL_CONTEXT
Zapisz kontekst po pobraniu komunikatu.

MQC.MQOO_SET
Otwórz, aby ustawić atrybuty-wymagane, aby ustawić właściwości.

MQC.MQOO_SET_ALL_CONTEXT
Umożliwia ustawienie całego kontekstu.

MQC.MQOO_SET_IDENTITY_CONTEXT
Umożliwia ustawienie kontekstu tożsamości.

QueueManagerName
Nazwa menedżera kolejek, w którym zdefiniowano kolejkę. Nazwa, która jest całkowicie pusta lub ma wartość NULL, oznacza menedżera kolejek, z którym połączony jest obiekt `MQQueueManager` .

dynamicQueueNazwa
Parametr *dynamicQueueName* jest ignorowany, chyba że parametr `queueName` określa nazwę kolejki modelowej. Jeśli tak, *dynamicQueueName* określa nazwę kolejki dynamicznej, która ma zostać utworzona. Pusta lub pusta nazwa nie jest poprawna, jeśli `queueName` określa nazwę kolejki modelowej. Jeśli ostatnim niepustym znakiem w nazwie jest gwiazdka (`*`), menedżer kolejek zastępuje gwiazdkę łańcuchem znaków. Znaki gwarantują, że nazwa wygenerowana dla kolejki jest unikalna w danym menedżerze kolejek.

Identyfikator alternateUser
Jeśli w parametrze `openOptions` określono parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` , parametr *alternateUserId* określa alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla operacji otwierania. Jeśli parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` nie zostanie określony, parametr *alternateUserId* może być pusty lub mieć wartość NULL.

public MQQueue(MQQueueManager *queueManager*, string *queueName*, int *openOptions*, string *queueManagerName*, string *dynamicQueueName*, string *alternateUserId*);

Zwraca wyjątek `MQException`.

Uzyskuje dostęp do kolejki w systemie `queueManager`.

Można pobrać lub przeglądać komunikaty, umieścić komunikaty, uzyskać informacje o atrybutach kolejki lub ustawić atrybuty kolejki. Jeśli kolejka o nazwie jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Wykonaj zapytanie o atrybut `name` wynikowego obiektu `MQQueue` , aby znaleźć nazwę kolejki dynamicznej.

queueManager
Menedżer kolejek, w którym ma zostać uzyskany dostęp do kolejki.

queueName
Nazwa kolejki do otwarcia.

openOptions
Opcje sterujące otwieraniem kolejki.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

MQC.MQOO_BIND_AS_QDEF
Użyj domyślnego powiązania dla kolejki.

MQC.MQOO_BIND_NOT_FIXED
Nie wiąż się z konkretnym miejscem docelowym.

MQC.MQOO_BIND_ON_OPEN
Powiąż uchwyt z miejscem docelowym, gdy kolejka jest otwarta.

MQC.MQOO_BROWSE
Otwórz, aby przeglądać komunikat.

MQC.MQOO_FAIL_IF_QUIESCING
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

MQC.MQOO_INPUT_AS_Q_DEF
Otwórz, aby pobrać komunikaty przy użyciu wartości domyślnej zdefiniowanej w kolejce.

MQC.MQOO_INPUT_SHARED
Otwórz, aby uzyskać wiadomości z dostępem współużytkowanym.

MQC.MQOO_INPUT_EXCLUSIVE
Otwórz, aby uzyskać wiadomości z wyłącznym dostępem.

MQC.MQOO_INQUIRE
Otwórz dla zapytania-wymagane, jeśli chcesz wykonać zapytanie o właściwości.

MQC.MQOO_OUTPUT
Otwórz, aby umieścić komunikaty.

MQC.MQOO_PASS_ALL_CONTEXT
Zezwalaj na przekazywanie całego kontekstu.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Zezwalaj na przekazywanie kontekstu tożsamości.

MQC.MQOO_SAVE_ALL_CONTEXT
Zapisz kontekst po pobraniu komunikatu.

MQC.MQOO_SET
Otwórz, aby ustawić atrybuty-wymagane, aby ustawić właściwości.

MQC.MQOO_SET_ALL_CONTEXT
Umożliwia ustawienie całego kontekstu.

MQC.MQOO_SET_IDENTITY_CONTEXT
Umożliwia ustawienie kontekstu tożsamości.

QueueManagerName
Nazwa menedżera kolejek, w którym zdefiniowano kolejkę. Nazwa, która jest całkowicie pusta lub ma wartość NULL, oznacza menedżera kolejek, z którym połączony jest obiekt `MQQueueManager` .

dynamicQueueNazwa
Parametr *dynamicQueueName* jest ignorowany, chyba że parametr `queueName` określa nazwę kolejki modelowej. Jeśli tak, *dynamicQueueName* określa nazwę kolejki dynamicznej, która ma zostać utworzona. Pusta lub pusta nazwa nie jest poprawna, jeśli `queueName` określa nazwę kolejki modelowej. Jeśli ostatnim niepustym znakiem w nazwie jest gwiazdka (`*`), menedżer kolejek zastępuje gwiazdkę łańcuchem znaków. Znaki gwarantują, że nazwa wygenerowana dla kolejki jest unikalna w danym menedżerze kolejek.

Identyfikator alternateUser
Jeśli w parametrze `openOptions` określono parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` , parametr *`alternateUserId`* określa alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla operacji otwierania. Jeśli parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` nie zostanie określony, parametr *`alternateUserId`* może być pusty lub mieć wartość NULL.

Klasa MQQueueManager.NET

Użyj programu `MQQueueManager` , aby nawiązać połączenie z menedżerem kolejek i uzyskać dostęp do obiektów menedżera kolejek. Steruje także transakcjami. Konstruktor `MQQueueManager` tworzy połączenie klienta lub serwera.

Klasa

```
System.Object
        │
        └─ IBM.WMQ.MQBase
                │
                └─ IBM.WMQ.MQBaseObject
                        │
                        └─ IBM.WMQ.ManagedObject
                                │
                                └─ IBM.WMQ.MQQueueManager
```

```
public class IBM.WMQ.MQQueueManager extends IBM.WMQ.MQManagedObject;
```

- “Właściwości” na stronie 1816
- “metody” na stronie 1819
- “Konstruktory” na stronie 1825

Właściwości

Test `MQException` zgłaszany podczas pobierania właściwości.

public int AccountingConnOverride {get;}
Określa, czy aplikacje mogą przesłaniać ustawienie wartości rozliczania i rozliczania kolejki `MQI` .

public int AccountingInterval {get;}
Czas przed zapisaniem pośrednich rekordów rozliczeniowych (w sekundach).

public int ActivityRecording {get;}
Steruje generowaniem raportów działania.

public int AdoptNewMCACheck {get;}
Określa, które elementy są sprawdzane w celu określenia, czy agent MCA jest adoptowany po wykryciu nowego kanału przychodzącego. Aby nazwa agenta MCA była adoptowana, musi być zgodna z nazwą aktywnego agenta MCA.

public int AdoptNewMCAInterval {get;}
Czas (w sekundach), przez który nowy kanał oczekuje na zakończenie osieroconego kanału.

public int AdoptNewMCAType {get;}
Określa, czy osierocona instancja MCA ma zostać adoptowana (zrestartowana) w przypadku wykrycia nowego żądania kanału przychodzącego zgodnego z wartością MCACheck AdoptNew.

public int BridgeEvent {get;}
Określa, czy są generowane zdarzenia mostu IMS .

public int ChannelEvent {get;}
Określa, czy generowane są zdarzenia kanału.

public int ChannelInitiatorControl {get;}
Określa, czy inicjator kanału jest uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

public int ChannelInitiatorAdapters {get;}
Liczba podzadań adaptera do przetworzenia wywołań IBM MQ .

public int ChannelInitiatorDispatchers {get;}
Liczba programów rozsyłających, które mają być używane dla inicjatora kanału.

public int ChannelInitiatorTraceAutoStart {get;}
Określa, czy śledzenie inicjatora kanału jest uruchamiane automatycznie.

public int ChannelInitiatorTraceTableSize {get;}
Wielkość (w megabajtach) obszaru danych śledzenia inicjatora kanału.

public int ChannelMonitoring {get;}
Określa, czy używane jest monitorowanie kanału.

public int ChannelStatistics {get;}
Steruje kolekcjonowaniem danych statystycznych dla kanałów.

public int CharacterSet {get;}
Zwraca identyfikator kodowanego zestawu znaków (CCSID) menedżera kolejek. Zestaw znaków `CharacterSet` jest używany przez menedżer kolejek dla wszystkich pól łańcucha znaków w aplikacyjnym interfejsie programistycznym.

public int ClusterSenderMonitoring {get;}
Steruje gromadzeniem danych monitorowania bezpośredniego dla automatycznie zdefiniowanych kanałów nadajnika klastra.

public int ClusterSenderStatistics {get;}
Steruje gromadzeniem danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastra.

public int ClusterWorkLoadMRU {get;}
Maksymalna liczba wychodzących kanałów klastra.

public int ClusterWorkLoadUseQ {get;}
Wartość domyślna właściwości `MQQueue` , `ClusterWorkLoadUseQ`, jeśli określono wartość `QMGR`.

public int CommandEvent {get;}
Określa, czy generowane są zdarzenia komend.

public string CommandInputQueueName {get;}
Zwraca nazwę kolejki wejściowej komend zdefiniowanej w menedżerze kolejek. Aplikacje mogą wysyłać komendy do tej kolejki, jeśli mają do tego uprawnienia.

public int CommandLevel {get;}
Wskazuje poziom funkcji menedżera kolejek. Zestaw funkcji, które odpowiadają określonemu poziomowi funkcji, zależy od platformy. Na konkretnej platformie można polegać na każdym menedżerze kolejek obsługującym funkcje na najniższym poziomie funkcjonalnym wspólnym dla wszystkich menedżerów kolejek.

public int CommandLevel {get;}
Określa, czy serwer komend jest uruchamiany automatycznie podczas uruchamiania menedżera kolejek.

public string DNSGroup {get;}
Nie jest już używany.

public int DNSWLM {get;}
Nie jest już używany.

public int IPAddressVersion {get;}
Protokół IP (IPv4 lub IPv6), który ma być używany dla połączenia kanału.

public boolean IsConnected {get;}
Zwraca wartość właściwości `isConnected`.

Wartość true oznacza, że połączenie z menedżerem kolejek zostało nawiązane i nie jest znane jako zerwane. Wszystkie wywołania funkcji IsConnected nie podejmują aktywnych prób nawiązania połączenia z menedżerem kolejek, dlatego możliwe jest, że połączenia fizyczne nie są nawiązywane, ale komenda IsConnected nadal zwraca wartość true (prawda). Stan IsConnected jest aktualizowany tylko wtedy, gdy w menedżerze kolejek wykonywane jest działanie, na przykład umieszczanie komunikatu, pobieranie komunikatu.

Jeśli ma wartość false, połączenie z menedżerem kolejek nie zostało nawiązane, zostało zerwane lub zostało rozłączone.

public int KeepAlive {get;}
Określa, czy narzędzie TCP KEEPALIVE ma być używane do sprawdzania, czy drugi koniec połączenia jest nadal dostępny. Jeśli jest niedostępny, kanał jest zamknięty.

public int ListenerTimer {get;}
Odstęp czasu (w sekundach) między próbami zrestartowania procesu nasłuchiwania przez program IBM MQ po awarii komunikacji APPC lub TCP/IP.

public int LoggerEvent {get;}
Określa, czy generowane są zdarzenia programu rejestrującego.

public string LU62ARMSuffix {get;}
Przyrostek elementu APPCPM systemu SYS1.PARMLIB. Przyrostek wyznacza LUADD do inicjatora kanału. Gdy menedżer automatycznego restartowania (ARM) restartuje inicjator kanału, wydawana jest komenda z/OS SET APPC=xx.

public string LUGroupName {get; z/os}
Ogólna nazwa jednostki logicznej, która ma być używana przez program nasłuchujący LU 6.2 obsługujący transmisje przychodzące dla grupy współużytkowania kolejek.

public string LUName {get;}
Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 .

public int MaximumActiveChannels {get;}
Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie.

public int MaximumCurrentChannels {get;}
Maksymalna liczba kanałów, które mogą być bieżące w dowolnym momencie (w tym kanały połączenia z serwerem z połączonymi klientami).

public int MaximumLU62Channels {get;}
Maksymalna liczba kanałów, które mogą być kanałami bieżącymi, lub klientów, które mogą być połączone, korzystających z protokołu transmisji LU 6.2 .

public int MaximumMessageLength {get;}
Zwraca maksymalną długość komunikatu (w bajtach), który może być obsłużony przez menedżera kolejek. Nie można zdefiniować kolejki z maksymalną długością komunikatu większą niż `MaximumMessageLength`.

public int MaximumPriority {get;}
Zwraca maksymalny priorytet komunikatu obsługiwany przez menedżera kolejek. Priorytety należą do zakresu od zera (najniższy) do tej wartości. Zgłasza wyjątek `MQException` , jeśli ta metoda zostanie wywołana po rozłączeniu się z menedżerem kolejek.

public int MaximumTCPChannels {get;}
Maksymalna liczba kanałów, które mogą być kanałami bieżącymi, lub klientów, które mogą być połączone, korzystających z protokołu transmisji TCP/IP.

public int MQIAccounting {get;}
Steruje kolekcjonowaniem informacji rozliczeniowych dla danych MQI.

public int MQIStatistics {get;}
Steruje kolekcjonowaniem informacji monitorowania statystyk dla menedżera kolejek.

public int OutboundPortMax {get;}
Maksymalna wartość z zakresu numerów portów, która ma być używana podczas wiązania kanałów wychodzących.

public int OutboundPortMin {get;}
Minimalna wartość z zakresu numerów portów, która ma być używana podczas wiązania kanałów wychodzących.

public int QueueAccounting {get;}
Określa, czy dane rozliczania klasy 3 (na poziomie wątku i na poziomie kolejki) mają być używane dla wszystkich kolejek.

public int QueueMonitoring {get;}
Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kolejek.

public int QueueStatistics {get;}
Steruje kolekcjonowaniem danych statystycznych dla kolejek.

public int ReceiveTimeout {get;}
Czas, przez jaki kanał TCP/IP oczekuje na odbiór danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego.

public int ReceiveTimeoutMin {get;}
Minimalny czas oczekiwania kanału TCP/IP na odbiór danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego.

public int ReceiveTimeoutType {get;}
Kwalifikator, który ma zostać zastosowany do wartości w polu `ReceiveTimeout`.

public int SharedQueueQueueManagerName {get;}
Określa sposób dostarczania komunikatów do kolejki współużytkowanej. Jeśli operacja put określa inny menedżer kolejek z tej samej grupy współużytkowania kolejek co docelowy menedżer kolejek, komunikat jest dostarczany na dwa sposoby:

MQC.MQSQQM_USE
Komunikaty są dostarczane do menedżera kolejek obiektów przed umieszczeniem w kolejce współużytkowanej.

MQCMQSQQM_IGNORE
Komunikaty są umieszczane bezpośrednio w kolejce współużytkowanej.

public int SSLEvent {get;}
Określa, czy są generowane zdarzenia TLS.

public int SSLFips {get;}
Określa, czy mają być używane tylko algorytmy z certyfikatem FIPS, jeśli szyfrowanie jest wykonywane w produkcie IBM MQ, a nie w sprzęcie szyfrującym.

public int SSLKeyResetCount {get;}
Wskazuje liczbę niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji TLS przed renegocjacją klucza tajnego.

public int ClusterSenderStatistics {get;}
Określa odstęp czasu (w minutach) między kolejnymi operacjami zbierania statystyk.

public int SyncpointAvailability {get;}
Wskazuje, czy menedżer kolejek obsługuje jednostki pracy i punkty synchronizacji przy użyciu metod `MQQueue.get` i `MQQueue.put` .

public string TCPName {get;}
Nazwa jedynego lub domyślnego systemu TCP/IP, który ma być używany, w zależności od wartości TCPStackType.

public int TCPStackType {get;}
Określa, czy inicjator kanału używa tylko przestrzeni adresowej TCP/IP określonej w parametrze `TCPName`. Alternatywnie inicjator kanału może być powiązany z dowolnym adresem TCP/IP.

public int TraceRouteRecording {get;}
Steruje rejestrowaniem informacji o śledzeniu trasy.

metody

public MQProcess AccessProcess(string *processName*, int *openOptions*);
public MQProcess AccessProcess(string *processName*, int *openOptions*, string *queueManagerName*, string *alternateUserId*);

Zwraca wyjątek `MQException`.

Uzyskaj dostęp do procesu IBM MQ w tym menedżerze kolejek, aby uzyskać informacje o atrybutach procesu.

processName
Nazwa procesu, który ma zostać otwarty.

openOptions
Opcje sterujące otwieraniem procesu. Poprawne opcje, które można dodać lub połączyć za pomocą bitowej operatora OR, to:

- `MQC.MQOO_FAIL_IF_QUIESCING`
- `MQC.MQOO_INQUIRE`

- MQC.MQOO_SET
- MQC.MQOO_ALTERNATE_USER_AUTHORITY

QueueManagerName
Nazwa menedżera kolejek, w którym jest zdefiniowany proces. Jeśli menedżer kolejek jest taki sam, jak menedżer, do którego proces uzyskuje dostęp, można pozostawić pustą lub pustą nazwę menedżera kolejek.

Identyfikator alternateUser

Jeśli parametr **openOptions** ma wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY , parametr *alternateUserId* określa alternatywny ID użytkownika używany do sprawdzania autoryzacji dla działania. Jeśli parametr MQOO_ALTERNATE_USER_AUTHORITY nie jest określony, parametr *alternateUserId* może być pusty lub mieć wartość null.

Jeśli nie określono opcji MQC.MQOO_ALTERNATE_USER_AUTHORITY , do połączenia z menedżerem kolejek używane jest domyślne uprawnienie użytkownika.

```
public MQQueue AccessQueue(string queueName, int openOptions);
public MQQueue AccessQueue(string queueName, int openOptions, string
queueManagerName, string dynamicQueueName, string alternateUserId);
```

Zwraca wyjątek MQException.

Uzyskuje dostęp do kolejki w tym menedżerze kolejek.

Można pobrać lub przeglądać komunikaty, umieścić komunikaty, uzyskać informacje o atrybutach kolejki lub ustawić atrybuty kolejki. Jeśli kolejka o nazwie jest kolejką modelową, tworzona jest dynamiczna kolejka lokalna. Wykonaj zapytanie o atrybut name wynikowego obiektu MQQueue , aby znaleźć nazwę kolejki dynamicznej.

queueName
Nazwa kolejki do otwarcia.

openOptions
Opcje sterujące otwieraniem kolejki.

MQC.MQOO_ALTERNATE_USER_AUTHORITY
Sprawdź poprawność przy użyciu podanego identyfikatora użytkownika.

MQC.MQOO_BIND_AS_QDEF
Użyj domyślnego powiązania dla kolejki.

MQC.MQOO_BIND_NOT_FIXED
Nie wiąż się z konkretnym miejscem docelowym.

MQC.MQOO_BIND_ON_OPEN
Powiąż uchwyt z miejscem docelowym, gdy kolejka jest otwarta.

MQC.MQOO_BROWSE
Otwórz, aby przeglądać komunikat.

MQC.MQOO_FAIL_IF_QUIESCING
Niepowodzenie, jeśli menedżer kolejek jest wyciszany.

MQC.MQOO_INPUT_AS_Q_DEF
Otwórz, aby pobrać komunikaty przy użyciu wartości domyślnej zdefiniowanej w kolejce.

MQC.MQOO_INPUT_SHARED
Otwórz, aby uzyskać wiadomości z dostępem współużytkowanym.

MQC.MQOO_INPUT_EXCLUSIVE
Otwórz, aby uzyskać wiadomości z wyłącznym dostępem.

MQC.MQOO_INQUIRE
Otwórz dla zapytania-wymagane, jeśli chcesz wykonać zapytanie o właściwości.

MQC.MQOO_OUTPUT
Otwórz, aby umieścić komunikaty.

MQC.MQOO_PASS_ALL_CONTEXT
Zezwalaj na przekazywanie całego kontekstu.

MQC.MQOO_PASS_IDENTITY_CONTEXT
Zezwalaj na przekazywanie kontekstu tożsamości.

MQC.MQOO_SAVE_ALL_CONTEXT
Zapisz kontekst po pobraniu komunikatu.

MQC.MQOO_SET
Otwórz, aby ustawić atrybuty-wymagane, aby ustawić właściwości.

MQC.MQOO_SET_ALL_CONTEXT
Umożliwia ustawienie całego kontekstu.

MQC.MQOO_SET_IDENTITY_CONTEXT
Umożliwia ustawienie kontekstu tożsamości.

QueueManagerName
Nazwa menedżera kolejek, w którym zdefiniowano kolejkę. Nazwa, która jest całkowicie pusta lub ma wartość NULL, oznacza menedżera kolejek, z którym połączony jest obiekt `MQQueueManager` .

dynamicQueueNazwa
Parametr *dynamicQueueName* jest ignorowany, chyba że parametr `queueName` określa nazwę kolejki modelowej. Jeśli tak, *dynamicQueueName* określa nazwę kolejki dynamicznej, która ma zostać utworzona. Pusta lub pusta nazwa nie jest poprawna, jeśli `queueName` określa nazwę kolejki modelowej. Jeśli ostatnim niepustym znakiem w nazwie jest gwiazdka (`*`), menedżer kolejek zastępuje gwiazdkę łańcuchem znaków. Znaki gwarantują, że nazwa wygenerowana dla kolejki jest unikalna w danym menedżerze kolejek.

Identyfikator alternateUser
Jeśli w parametrze `openOptions` określono parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` , parametr *`alternateUserId`* określa alternatywny identyfikator użytkownika, który jest używany do sprawdzania autoryzacji dla operacji otwierania. Jeśli parametr `MQC.MQOO_ALTERNATE_USER_AUTHORITY` nie zostanie określony, parametr *`alternateUserId`* może być pusty lub mieć wartość NULL.

```
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName);
public MQTopic AccessTopic( MQDestination destination, string topicName, string
topicObject, int options, string alternateUserId, string subscriptionName,
System.Collections.Hashtable properties);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options);
public MQTopic AccessTopic(string topicName, string topicObject, int openAs,
int options, string alternateUserId);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName);
public MQTopic AccessTopic(string topicName, string topicObject, int options,
string alternateUserId, string subscriptionName, System.Collections.Hashtable
properties);
```

Uzyskaj dostęp do tematu w tym menedżerze kolejek.

Obiekty `MQTopic` są ściśle powiązane z obiektami tematów administracyjnych, które są czasami nazywane obiektami tematów. Na wejściu `topicObject` wskazuje obiekt tematu administracyjnego. Konstruktor `MQTopic` uzyskuje łańcuch tematu z obiektu tematu i łączy go z nazwą `topicName` w celu utworzenia nazwy tematu. Parametr `topicObject` lub `topicName` może mieć wartość NULL. Nazwa tematu jest dopasowywana do drzewa tematów, a nazwa najbardziej zgodnego obiektu tematu administracyjnego jest zwracana w sekcji `topicObject`.

Tematy powiązane z obiektem MQTopic są wynikiem połączenia dwóch łańcuchów tematów. Pierwszy łańcuch tematu jest definiowany przez obiekt tematu administracyjnego identyfikowany przez *topicObject*. Drugi łańcuch tematu to *topicString*. Wynikowy łańcuch tematu powiązany z obiektem MQTopic może identyfikować wiele tematów, w tym znaki wieloznaczne.

W zależności od tego, czy temat jest otwarty do publikowania lub subskrybowania, można użyć metod MQTopic.Put do publikowania tematów lub metod MQTopic.Get do odbierania publikacji na tematy. Aby publikować i subskrybować ten sam temat, należy uzyskać do niego dostęp dwa razy, raz dla publikowania i raz dla subskrybowania.

W przypadku tworzenia obiektu MQTopic dla subskrypcji bez udostępniania obiektu MQDestination przyjmowana jest subskrypcja zarządzana. Jeśli kolejka zostanie przekazana jako obiekt MQDestination , zostanie przyjęta niezarządzana subskrypcja. Należy upewnić się, że ustawione opcje subskrypcji są spójne z zarządzaną lub niezarządzaną subskrypcją.

miejsce docelowe
destination jest instancją MQQueue . Podanie wartości *destination*powoduje, że program MQTopic jest otwierany jako niezarządzana subskrypcja. Publikacje w tym temacie są dostarczane do kolejki, do której dostęp jest uzyskiwany jako *destination*.

topicName
Łańcuch tematu, który jest drugą częścią nazwy tematu. Łańcuch *topicName* jest konkatenowany z łańcuchem tematu zdefiniowanym w obiekcie tematu administracyjnego produktu *topicObject* . Parametr *topicName* można ustawić na wartość NULL. W takim przypadku nazwa tematu jest definiowana przez łańcuch tematu w pliku *topicObject*.

topicObject
Na wejściu *topicObject* jest nazwą obiektu tematu zawierającego łańcuch tematu, który stanowi pierwszą część nazwy tematu. Łańcuch tematu w pliku *topicObject* jest konkatenowany z łańcuchem *topicName*. Reguły tworzenia łańcuchów tematów są zdefiniowane w sekcji Łączenie łańcuchów tematów.

W danych wyjściowych *topicObject* zawiera nazwę obiektu tematu administracyjnego, który jest najbardziej zgodny w drzewie tematów z tematem identyfikowanym przez łańcuch tematu.

openAs
Dostęp do tematu w celu publikowania lub subskrybowania. Parametr może zawierać tylko jedną z następujących opcji:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opcje

Połącz opcje sterujące otwieraniem tematu dla publikacji lub subskrypcji. Stałe MQC.MQSO_* umożliwiają dostęp do tematu dla subskrypcji, a stałe MQC.MQOO_* umożliwiają dostęp do tematu dla publikacji.

Jeśli wymagana jest więcej niż jedna opcja, dodaj wartości lub połącz wartości opcji przy użyciu operatora bitowej OR .

Identyfikator alternateUser
Podaj alternatywny ID użytkownika, który jest używany do sprawdzania wymaganych uprawnień do zakończenia operacji. Jeśli w parametrze opcji ustawiono wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY lub MQC.MQSO_ALTERNATE_USER_AUTHORITY , należy podać wartość *alternateUserId*.

subscriptionName
Parametr *subscriptionName* jest wymagany, jeśli podano opcje MQC.MQSO_DURABLE lub MQC.MQSO_ALTER . W obu przypadkach program MQTopic jest niejawnie otwierany do subskrypcji. Wyjątek jest zgłaszany, jeśli właściwość MQC.MQSO_DURABLE jest ustawiona, a subskrypcja istnieje lub jeśli właściwość MQC.MQSO_ALTER jest ustawiona, a subskrypcja nie istnieje.

Właściwości
Ustaw dowolną ze specjalnych właściwości subskrypcji wymienionych przy użyciu tabeli mieszającej. Podane pozycje w tabeli mieszającej są aktualizowane wartościami wyjściowymi. Pozycje nie są dodawane do tabeli mieszającej w celu raportowania wartości wyjściowych.

- 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();

Zwraca wyjątek MQException

Zwraca obiekt MQAsyncStatus reprezentujący działanie asynchroniczne połączenia menedżera kolejek.

public void Backout();

Zwraca wyjątek MQException.

Wycofuje wszystkie komunikaty, które zostały odczytane lub zapisane w punkcie synchronizacji od ostatniego punktu synchronizacji.

Komunikaty, które zostały zapisane z ustawioną flagą MQC.MQPMO_SYNCPOINT , są usuwane z kolejek. Komunikaty odczytane z flagą MQC.MQGMO_SYNCPOINT są przywracane do kolejek, z których pochodzą. Jeśli komunikaty są trwałe, zmiany są rejestrowane.

W przypadku klientów z możliwością ponownego połączenia kod przyczyny MQRC_NONE jest zwracany do klienta po pomyślnym ponownym nawiązaniu połączenia.

public void Begin();

Zwraca wyjątek MQException.

Opcja Begin jest obsługiwana tylko w trybie powiązań serwera. Uruchamia globalną jednostkę pracy.

public void Commit();

Zwraca wyjątek MQException.

Zatwierdź wszystkie komunikaty, które zostały odczytane lub zapisane w punkcie synchronizacji od ostatniego punktu synchronizacji.

Komunikaty zapisane z ustawioną flagą MQC.MQPMO_SYNCPOINT są udostępniane innym aplikacjom. Komunikaty pobrane z ustawioną flagą MQC.MQGMO_SYNCPOINT są usuwane. Jeśli komunikaty są trwałe, zmiany są rejestrowane.

Do klienta z możliwością ponownego połączenia zwracane są następujące kody przyczyny:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało utracone podczas wykonywania wywołania zatwierdzenia.
- MQRC_BACKED_OUT , jeśli wywołanie zatwierdzenia zostało wydane po ponownym połączeniu.

Disconnect();

Zwraca wyjątek MQException.

Zamknij połączenie z menedżerem kolejek. Wszystkie obiekty, do których uzyskano dostęp w tym menedżerze kolejek, nie są już dostępne dla tej aplikacji. Aby ponownie uzyskać dostęp do obiektów, należy utworzyć obiekt MQQueueManager .

Ogólnie rzecz biorąc, każda praca wykonywana w ramach jednostki pracy jest zatwierdzana. Jeśli jednak jednostka pracy jest zarządzana przez produkt .NET, jednostka pracy może zostać wycofana.

```
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);
```

Zwraca wyjątek `MQException`.

Umieszcza pojedynczy komunikat w kolejce lub temacie bez uprzedniego tworzenia obiektu `MQQueue` lub `MQTopic` .

queueName
Nazwa kolejki, w której ma zostać umieszczony komunikat.

destinationName
Nazwa obiektu docelowego. Jest to kolejka lub temat w zależności od wartości parametru *type*.

typ
Typ obiektu docelowego. Nie można łączyć opcji.

MQC.MQOT_Q
Kolejka

MQC.MQOT_TOPIC
Temat

QueueManagerName

Nazwa menedżera kolejek lub alias menedżera kolejek, w którym zdefiniowano kolejkę. Jeśli określono typ `MQC.MQOT_TOPIC` , ten parametr jest ignorowany.

Jeśli kolejka jest kolejką modelową, a rozstrzygnięta nazwa menedżera kolejek nie jest nazwą tego menedżera kolejek, zgłaszany jest wyjątek `MQException` .

topicString

topicString jest połączona z nazwą tematu w obiekcie tematu *destinationName* .

Parametr *topicString* jest ignorowany, jeśli parametr *destinationName* jest kolejką.

komunikat
Komunikat do wysłania. Komunikat jest obiektem wejścia/wyjścia.

Do klienta z możliwością ponownego połączenia zwracane są następujące kody przyczyny:

- `MQRC_CALL_INTERRUPTED` , jeśli połączenie zostało zerwane podczas wykonywania wywołania metody Put dla komunikatu trwałego.
- `MQRC_NONE` , jeśli połączenie zostało pomyślnie nawiązane podczas wykonywania wywołania metody Put dla komunikatu nietrwałego (patrz sekcja Odtwarzanie aplikacji).

Opcje komendy putMessage

Opcje sterujące działaniami operacji umieszczania.

Jeśli parametr *putMessageOptions*zostanie pominięty, zostanie utworzona domyślna instancja parametru *putMessageOptions* . *putMessageOptions* jest obiektem wejścia/wyjścia.

Jeśli opcja MQPMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

Identyfikator alternateUser

Określa alternatywny identyfikator użytkownika używany do sprawdzania autoryzacji podczas umieszczania komunikatu w kolejce.

Parametr *alternateUserId* można pominąć, jeśli nie ustawiono parametru MQC.MQOO_ALTERNATE_USER_AUTHORITY w pliku *putMessageOptions*. Jeśli zostanie ustawiona wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY, należy również ustawić wartość *alternateUserId*. Parametr *alternateUserId* działa tylko wtedy, gdy ustawiono również parametr MQC.MQOO_ALTERNATE_USER_AUTHORITY.

Konstruktory

```
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);
```

Zwraca wyjątek MQException.

Tworzy połączenie z menedżerem kolejek. Wybierz między tworzeniem połączenia z klientem a połączeniem z serwerem.

Podczas próby nawiązania połączenia z menedżerem kolejek wymagane jest uprawnienie do wykonywania zapytania (inq) w menedżerze kolejek. Próba nawiązania połączenia nie powiedzie się bez uprawnienia do uzyskiwania informacji.

Połączenie klienckie jest tworzone, jeśli spełniony jest jeden z następujących warunków:

1. *channel* lub *connName* są określone w konstruktorze.
2. *HostName*, *Port*lub *Channel* są określone w pliku *properties*.
3. Określono *MQEnvironment.HostName*, *MQEnvironment.Port*lub *MQEnvironment.Channel* .

Wartości właściwości połączenia są ustawiane domyślnie w podanej kolejności. Wartości *channel* i *connName* w konstruktorze mają pierwszeństwo przed wartościami właściwości w konstruktorze. Wartości właściwości konstruktora mają pierwszeństwo właściwości MQEnvironment .

Nazwa hosta, nazwa kanału i port są zdefiniowane w klasie MQEnvironment .

QueueManagerName
Nazwa menedżera kolejek lub grupy menedżerów kolejek, z którymi ma zostać nawiązane połączenie.

Aby wybrać domyślny menedżer kolejek, pomiń parametr lub pozostaw wartość NULL albo pozostaw wartość pustą. Domyślne połączenie menedżera kolejek na serwerze jest nawiązywane z domyślnym menedżerem kolejek na serwerze. Domyślne połączenie menedżera kolejek w połączeniu klienta jest nawiązywane z menedżerem kolejek, z którym jest połączony program nasłuchujący.

opcje
Określ opcje połączenia MQCNO . Wartości muszą mieć zastosowanie do typu tworzonego połączenia. Jeśli na przykład zostaną określone następujące właściwości połączenia z serwerem dla połączenia klienta, zostanie zgłoszony wyjątek MQException .

- MQC.MQCNO_FASTPATH_BINDING
- MQC.MQCNO_STANDARD_BINDING

Właściwości

Parametr properties przyjmuje serię par klucz/wartość, które przesłaniają właściwości ustawione przez `MQEnvironment` ; Patrz przykład: “Przesłoń właściwości MQEnvironment” na stronie 1828. Można przesłonić następujące właściwości:

- `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`

kanał

Nazwa kanału połączenia z serwerem

connName

Nazwa połączenia w formacie *HostName* (*Port*).

Listę *nazw hostów* i *portów* można podać jako argument konstruktora `MQQueueManager (String queueManagerName, Hashtable properties)` przy użyciu właściwości CONNECTION_NAME_PROPERTY.

Na przykład:

```
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);
```

Podczas próby nawiązania połączenia lista nazw połączeń jest przetwarzana w kolejności. Jeśli próba nawiązania połączenia z pierwszą nazwą hosta i portem nie powiedzie się, podejmowana jest próba nawiązania połączenia z drugą parą atrybutów. Klient powtarza ten proces do momentu pomyślnego nawiązania połączenia lub wyczerpania listy. Jeśli lista zostanie wyczerpana, do aplikacji klienckiej zostanie zwrócony odpowiedni kod przyczyny i kod zakończenia.

Jeśli dla nazwy połączenia nie podano numeru portu, jest to port domyślny (skonfigurowany w pliku `mqclient.ini`). jest używany.

Ustaw listę połączeń

Listę połączeń można ustawić przy użyciu następujących metod, gdy ustawione są opcje automatycznego ponownego połączenia klienta:

Ustawianie listy połączeń za pośrednictwem MQSERVER

Listę połączeń można ustawić w wierszu komend.

W wierszu komend ustaw następującą komendę:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/Hostname1(Port1),Hostname2(Por2),Hostname3(Port3)
```

Na przykład:

```
MQSERVER=SYSTEM.DEF.SVRCONN/TCP/fred.mq.com(5266),nick.mq.com(6566),jack.mq.com(8413)
```

Jeśli połączenie zostanie ustawione w serwerze MQSERVER, nie należy ustawiać go w aplikacji.

Jeśli lista połączeń zostanie ustawiona w aplikacji, aplikacja nadpisze wszystko, co jest ustawione w zmiennej środowiskowej MQSERVER.

Ustawianie listy połączeń za pośrednictwem aplikacji

Listę połączeń można ustawić w aplikacji, określając nazwę hosta i właściwości portu.

```
String connName = "fred.mq.com(2344), nick.mq.com(3746), chris.mq.com(4288)";
MQQueueManager qm = new MQQueueManager("QM1", "TestChannel", connName);
```

Ustawianie listy połączeń za pomocą programu `app.config`

`App.config` to plik XML, w którym określane są pary klucz-wartość.

Na liście połączeń podaj

```
<app.Settings>
<add key="Connection1" value="Hostname1(Port1)"/>
<add key="Connection2" value="Hostname2(Port2)"/>
<app.Settings>
```

Na przykład:

```
<app.Settings>
<add key>="Connection1" value="fred.mq.com(2966)"/>
<add key>="Connection2" value="alex.mq.com(6533)"/>
<app.Settings>
```

Listę połączeń można zmienić bezpośrednio w pliku `app.config` .

Ustawianie listy połączeń za pomocą programu `MQEnvironment`

Aby ustawić listę połączeń za pośrednictwem `MQEnvironment`, należy użyć właściwości *ConnectionName* .

```
MQEnvironment.ConnectionName = "fred.mq.com(4288),"alex.mq.com(5211);
```

Właściwość *ConnectionName* zastępuje właściwości nazwy hosta i portu ustawione w pliku `MQEnvironment`.

Utwórz połączenie klienta

W poniższym przykładzie przedstawiono sposób tworzenia połączenia klienta z menedżerem kolejek. Połączenie klienta można utworzyć, ustawiając zmienne `MQEnvironment` przed utworzeniem nowego obiektu `MQQueueManager` .

```
MQEnvironment.Hostname = "fred.mq.com";  // host to connect to
MQEnvironment.Port     = 1414;           // port to connect to
                                         //If not explicitly set,
                                         // defaults to 1414
                                         // (the default IBM MQ port)
MQEnvironment.Channel  = "channel.name"; // the case sensitive
                                         //  name of the
                                         // SVR CONN channel on
                                         // the queue manager
MQQueueManager qMgr    = new MQQueueManager("MYQM");
```

Rysunek 11. Połączenie klienta

Przesłoń właściwości `MQEnvironment`

Poniższy przykład przedstawia sposób tworzenia menedżera kolejek z identyfikatorem użytkownika i hasłem zdefiniowanym w tabeli mieszającej.

```
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);
}
```

Rysunek 12. Przesłanianie właściwości `MQEnvironment`

Utwórz połączenie z możliwością ponownego połączenia

W poniższym przykładzie przedstawiono sposób automatycznego ponownego połączenia klienta z menedżerem kolejek.

```
Hashtable properties = new Hashtable(); // The queue manager name and the
                                 // properties how it has to be connected

properties.Add(MQC.CONNECT_OPTIONS_PROPERTY, MQC.MQCNO_RECONNECT); // Options
                                 // through which reconnection happens

properties.Add(MQC.CONNECTION_NAME_PROPERTY,"fred.mq.com(4789),nick.mq.com(4790)"); // The list
                                 // of queue managers through which reconnection happens

MQ QueueManager qmgr = new MQQueueManager("qmgrname", properties);
```

Rysunek 13. Automatyczne ponowne łączenie klienta z menedżerem kolejek

Klasa `MQSubscription.NET`

Użyj opcji `MQSubscription` , aby zażądać, aby zachowane publikacje zostały wysłane do subskrybenta. `MQSubscription` jest właściwością obiektu `MQTopic` otwartego dla subskrypcji.

Klasa


```
public class IBM.WMQ.MQSubscription extends IBM.WMQ.MQManagedObject;
```

- “Właściwości” na stronie 1829
- “metody” na stronie 1829
- “Konstruktory” na stronie 1829

Właściwości

Uzyskaj dostęp do właściwości subskrypcji za pomocą klasy `MQManagedObject` ; patrz sekcja “Właściwości” na stronie 1787.

metody

Uzyskaj dostęp do metod `Inquire`, `Set` i `Get` subskrypcji przy użyciu klasy `MQManagedObject` (patrz sekcja “metody” na stronie 1787).

public int RequestPublicationUpdate(int *options*);

Zwraca wyjątek `MQException`.

Zażądaj zaktualizowanej publikacji dla bieżącego tematu. Jeśli menedżer kolejek ma zachowane publikacje dla tematu, są one wysyłane do subskrybenta.

Przed wywołaniem metody `RequestPublicationUpdate`należy otworzyć temat dla subskrypcji, aby uzyskać obiekt `MQSubscription` .

Zwykle należy otworzyć subskrypcję za pomocą opcji `MQC.MQSO_PUBLICATIONS_ON_REQUEST` . Jeśli w łańcuchu tematu nie ma znaków wieloznacznych, w wyniku tego wywołania wysyłana jest tylko jedna publikacja. Jeśli łańcuch tematu zawiera znaki wieloznaczne, może zostać wysłanych wiele publikacji. Ta metoda zwraca liczbę zachowanych publikacji, które są wysyłane do kolejki subskrypcji. Nie ma gwarancji, że taka liczba publikacji zostanie odebrana, szczególnie jeśli są to komunikaty nietrwałe.

opcje

MQC.MQSRO_FAIL_IF_QUIESCING
Wykonanie metody nie powiedzie się, jeśli menedżer kolejek jest w stanie wyciszenia. W systemie z/OS, dla aplikacji CICS lub IMS , `MQC.MQSRO_FAIL_IF_QUIESCING` również wymusza niepowodzenie metody, jeśli połączenie jest w stanie wyciszenia.

MQC.MQSRO_NONE
Nie określono żadnych opcji.

Konstruktory

Brak konstruktora Public .

Obiekt `MQSubscription` jest zwracany we właściwości `SubscriptionReference` obiektu `MQTopic` , który jest otwarty dla subskrypcji.

Wywołaj metodę `RequestPublicationUpdate` . `MQSubscription` jest podklasą klasy `MQManagedObject`. Użyj odwołania, aby uzyskać dostęp do właściwości i metod `MQManagedObject`.

Klasa MQTopic.NET

Produkt MQTopic służy do publikowania i subskrybowania komunikatów w temacie oraz do tworzenia zapytań i ustawiania atrybutów tematu. Utwórz obiekt MQTopic na potrzeby publikowania lub subskrybowania przy użyciu konstruktora lub metody MQQueueManager.AccessTopic .

Klasa

```
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;
```

- “Właściwości” na stronie 1830
- “metody” na stronie 1830
- “Konstruktory” na stronie 1832

Właściwości

Test MQException zgłaszany podczas pobierania właściwości.

public Boolean IsDurable {get;}
Właściwość tylko do odczytu, która zwraca wartość True , jeśli subskrypcja jest trwała lub wartość False w przeciwnym razie. Jeśli temat został otwarty do publikacji, właściwość jest ignorowana i zawsze zwraca wartość False.

public Boolean IsManaged {get;};
Właściwość tylko do odczytu, która zwraca wartość True , jeśli subskrypcja jest zarządzana przez menedżer kolejek lub wartość False w przeciwnym razie. Jeśli temat został otwarty do publikacji, właściwość jest ignorowana i zawsze zwraca wartość False.

public Boolean IsSubscribed {get;};
Właściwość tylko do odczytu, która zwraca wartość True , jeśli temat został otwarty dla subskrypcji, i wartość False , jeśli temat został otwarty dla publikacji.

public MQSubscription SubscriptionReference {get;};
Właściwość tylko do odczytu, która zwraca obiekt MQSubscription powiązany z obiektem tematu otwartym dla subskrypcji. Odwołanie jest dostępne, jeśli użytkownik chce zmodyfikować opcje zamykania lub uruchomić dowolną z metod obiektów.

public MQDestination UnmanagedDestinationReference {get;};
Właściwość tylko do odczytu, która zwraca wartość MQQueue powiązaną z niezarządzaną subskrypcją. Jest to miejsce docelowe określone podczas tworzenia obiektu tematu. Ta właściwość zwraca wartość NULL dla wszystkich obiektów tematu otwartych do publikacji lub z subskrypcją zarządzaną.

metody

public void Put(MQMessage *message*);
public void Put(MQMessage *message*, MQPutMessageOptions *putMessageOptions*);
Zgłasza wyjątek MQException.

Publikuje komunikat w temacie.

Modyfikacje obiektu MQMessage dokonane po wywołaniu metody Put nie mają wpływu na rzeczywisty komunikat w kolejce produktu IBM MQ lub w temacie publikacji.

Put aktualizuje właściwości MessageId i CorrelationId obiektu MQMessage i nie czyści danych komunikatu. Dalsze wywołania Put lub Get odnoszą się do zaktualizowanych informacji w obiekcie MQMessage . Na przykład w poniższym fragmencie kodu pierwszy komunikat zawiera a , a drugi ab.

```
msg.WriteString("a");
q.Put(msg,pmo);
msg.WriteString("b");
q.Put(msg,pmo);
```

komunikat
Obiekt MQMessage zawierający dane deskryptora komunikatu i komunikat do wysłania. W wyniku tej metody można zmienić deskryptor komunikatu. Wartości w deskryptorze komunikatu bezpośrednio po zakończeniu tej metody są wartościami, które zostały umieszczone w kolejce lub opublikowane w temacie.

Do klienta z możliwością ponownego połączenia zwracane są następujące kody przyczyny:

- MQRC_CALL_INTERRUPTED , jeśli połączenie zostało zerwane podczas wykonywania wywołania metody Put dla trwałego komunikatu i ponowne połączenie zakończyło się pomyślnie.
- MQRC_NONE , jeśli połączenie zostało pomyślnie nawiązane podczas wykonywania wywołania metody Put dla komunikatu nietrwałego (patrz sekcja Odtwarzanie aplikacji).

Opcje komendy putMessage
Opcje sterujące działaniem operacji put.

Jeśli parametr *putMessageOptions* nie zostanie podany, zostanie użyta domyślna instancja MQPutMessageOptons .

Jeśli opcja MQPMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

Uwaga: W celu uproszczenia i zwiększenia wydajności, jeśli pojedynczy komunikat ma zostać umieszczony w kolejce, należy użyć obiektu MQQueueManager.Put . W tym celu powinien istnieć obiekt MQQueue .

public void Get(MQMessage *message*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*);
public void Get(MQMessage *message*, MQGetMessageOptions *getMessageOptions*, int *MaxMsgSize*);
Zgłasza wyjątek MQException.

Pobiera komunikat z tematu.

W tej metodzie do pobrania używana jest domyślna instancja metody MQGetMessageOptions . Używana jest opcja komunikatu MQGMO_NOWAIT.

Jeśli operacja pobierania nie powiedzie się, obiekt MQMessage pozostanie niezmieniony. Jeśli operacja powiedzie się, części deskryptora komunikatu i danych komunikatu w pliku MQMessage są zastępowane deskryptorem komunikatu i danymi komunikatu z komunikatu przychodzącego.

Wszystkie wywołania IBM MQ z określonego MQQueueManager są synchroniczne. Z tego powodu, jeśli wykonywana jest operacja get with wait, wszystkie inne wątki korzystające z tego samego MQQueueManager będą miały zablokowaną możliwość wykonywania kolejnych wywołań IBM MQ do czasu wykonania wywołania Get. Jeśli dostęp do programu IBM MQ ma być uzyskiwany jednocześnie z wielu wątków, każdy wątek musi utworzyć własny obiekt MQQueueManager .

komunikat
Zawiera deskryptor komunikatu i zwracane dane komunikatu. Niektóre pola w deskryptorze komunikatu są parametrami wejściowymi. Należy upewnić się, że parametry wejściowe MessageId i CorrelationId są ustawione zgodnie z wymaganiami.

Klient z możliwością ponownego połączenia zwraca kod przyczyny MQRC_BACKED_OUT po pomyślnym ponownym nawiązaniu połączenia dla komunikatów odebranych w ramach elementu MQGM_SYNCPOINT.

Opcje getMessage
Opcje sterujące działaniem pobierania.

Użycie opcji MQC.MQGMO_CONVERT może spowodować wystąpienie wyjątku z kodem przyczyny MQC.MQRC_CONVERTED_STRING_TOO_BIG podczas przekształcania kodów znaków jednobajtowych w kody dwubajtowe. W takim przypadku komunikat jest kopiowany do buforu bez konwersji.

Jeśli parametr *getMessageOptions* nie jest określony, używana jest opcja komunikatu MQGMO_NOWAIT.

Jeśli opcja MQGMO_LOGICAL_ORDER jest używana w kliencie z możliwością ponownego połączenia, zwracany jest kod przyczyny MQRC_RECONNECT_INCOMPATIBLE .

MaxMsg
Największy komunikat, który ma zostać odebrany przez ten obiekt komunikatu. Jeśli komunikat w kolejce jest większy niż ta wielkość, wystąpi jedna z dwóch sytuacji:

- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG jest ustawiona w obiekcie MQGetMessageOptions , komunikat jest wypełniany możliwie największą ilością danych komunikatu. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_ACCEPTED .
- Jeśli opcja MQGMO_ACCEPT_TRUNCATED_MSG nie jest ustawiona, komunikat pozostaje w kolejce. Zgłaszany jest wyjątek z kodem zakończenia MQCC_WARNING i kodem przyczyny MQRC_TRUNCATED_MSG_FAILED .

Jeśli parametr *MaxMsgSize* nie jest określony, pobierany jest cały komunikat.

Konstruktory

```
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);
```

Dostęp do tematu w serwisie *queueManager*.

Obiekty MQTopic są ściśle powiązane z obiektami tematów administracyjnych, które są czasami nazywane obiektami tematów. Na wejściu topicObject wskazuje obiekt tematu administracyjnego. Konstruktor MQTopic uzyskuje łańcuch tematu z obiektu tematu i łączy go z nazwą topicName w celu utworzenia nazwy tematu. Parametr topicObject lub topicName może mieć wartość NULL. Nazwa tematu jest dopasowywana do drzewa tematów, a nazwa najbardziej zgodnego obiektu tematu administracyjnego jest zwracana w sekcji topicObject.

Tematy powiązane z obiektem MQTopic są wynikiem połączenia dwóch łańcuchów tematów. Pierwszy łańcuch tematu jest definiowany przez obiekt tematu administracyjnego identyfikowany przez *topicObject*. Drugi łańcuch tematu to *topicString*. Wynikowy łańcuch tematu powiązany z obiektem MQTopic może identyfikować wiele tematów, w tym znaki wieloznaczne.

W zależności od tego, czy temat jest otwarty do publikowania lub subskrybowania, można użyć metod MQTopic.Put do publikowania tematów lub metod MQTopic.Get do odbierania publikacji na tematy. Aby publikować i subskrybować ten sam temat, należy uzyskać do niego dostęp dwa razy, raz dla publikowania i raz dla subskrybowania.

W przypadku tworzenia obiektu MQTopic dla subskrypcji bez udostępniania obiektu MQDestination przyjmowana jest subskrypcja zarządzana. Jeśli kolejka zostanie przekazana jako obiekt MQDestination , zostanie przyjęta niezarządzana subskrypcja. Należy upewnić się, że ustawione opcje subskrypcji są spójne z zarządzaną lub niezarządzaną subskrypcją.

queueManager
Menedżer kolejek, dla którego ma zostać uzyskany dostęp do tematu.

miejsce docelowe
destination jest instancją MQQueue . Podanie wartości *destination*powoduje, że program MQTopic jest otwierany jako niezarządzana subskrypcja. Publikacje w tym temacie są dostarczane do kolejki, do której dostęp jest uzyskiwany jako *destination*.

topicName
Łańcuch tematu, który jest drugą częścią nazwy tematu. Łańcuch *topicName* jest konkatenowany z łańcuchem tematu zdefiniowanym w obiekcie tematu administracyjnego produktu *topicObject* . Parametr *topicName* można ustawić na wartość NULL. W takim przypadku nazwa tematu jest definiowana przez łańcuch tematu w pliku *topicObject*.

topicObject
Na wejściu *topicObject* jest nazwą obiektu tematu zawierającego łańcuch tematu, który stanowi pierwszą część nazwy tematu. Łańcuch tematu w pliku *topicObject* jest konkatenowany z łańcuchem *topicName*. Reguły tworzenia łańcuchów tematów są zdefiniowane w sekcji Łączenie łańcuchów tematów.

W danych wyjściowych *topicObject* zawiera nazwę obiektu tematu administracyjnego, który jest najbardziej zgodny w drzewie tematów z tematem identyfikowanym przez łańcuch tematu.

openAs
Dostęp do tematu w celu publikowania lub subskrybowania. Parametr może zawierać tylko jedną z następujących opcji:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opcje

Połącz opcje sterujące otwieraniem tematu dla publikacji lub subskrypcji. Stałe MQC.MQSO_* umożliwiają dostęp do tematu dla subskrypcji, a stałe MQC.MQOO_* umożliwiają dostęp do tematu dla publikacji.

Jeśli wymagana jest więcej niż jedna opcja, dodaj wartości lub połącz wartości opcji przy użyciu operatora bitowej OR .

Identyfikator alternateUser
Podaj alternatywny ID użytkownika, który jest używany do sprawdzania wymaganych uprawnień do zakończenia operacji. Jeśli w parametrze opcji ustawiono wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY lub MQC.MQSO_ALTERNATE_USER_AUTHORITY , należy podać wartość *alternateUserId*.

subscriptionName
Parametr *subscriptionName* jest wymagany, jeśli podano opcje MQC.MQSO_DURABLE lub MQC.MQSO_ALTER . W obu przypadkach program MQTopic jest niejawnie otwierany do subskrypcji. Wyjątek jest zgłaszany, jeśli właściwość MQC.MQSO_DURABLE jest ustawiona,

a subskrypcja istnieje lub jeśli właściwość MQC.MQSO_ALTER jest ustawiona, a subskrypcja nie istnieje.

Właściwości

Ustaw dowolną ze specjalnych właściwości subskrypcji wymienionych przy użyciu tabeli mieszającej. Podane pozycje w tabeli mieszającej są aktualizowane wartościami wyjściowymi. Pozycje nie są dodawane do tabeli mieszającej w celu raportowania wartości wyjściowych.

- 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*);

Uzyskaj dostęp do tematu w tym menedżerze kolejek.

Obiekty MQTopic są ściśle powiązane z obiektami tematów administracyjnych, które są czasami nazywane obiektami tematów. Na wejściu topicObject wskazuje obiekt tematu administracyjnego. Konstruktor MQTopic uzyskuje łańcuch tematu z obiektu tematu i łączy go z nazwą topicName w celu utworzenia nazwy tematu. Parametr topicObject lub topicName może mieć wartość NULL. Nazwa tematu jest dopasowywana do drzewa tematów, a nazwa najbardziej zgodnego obiektu tematu administracyjnego jest zwracana w sekcji topicObject.

Tematy powiązane z obiektem MQTopic są wynikiem połączenia dwóch łańcuchów tematów. Pierwszy łańcuch tematu jest definiowany przez obiekt tematu administracyjnego identyfikowany przez *topicObject*. Drugi łańcuch tematu to *topicString*. Wynikowy łańcuch tematu powiązany z obiektem MQTopic może identyfikować wiele tematów, w tym znaki wieloznaczne.

W zależności od tego, czy temat jest otwarty do publikowania lub subskrybowania, można użyć metod MQTopic.Put do publikowania tematów lub metod MQTopic.Get do odbierania publikacji na tematy. Aby publikować i subskrybować ten sam temat, należy uzyskać do niego dostęp dwa razy, raz dla publikowania i raz dla subskrybowania.

W przypadku tworzenia obiektu MQTopic dla subskrypcji bez udostępniania obiektu MQDestination przyjmowana jest subskrypcja zarządzana. Jeśli kolejka zostanie przekazana jako obiekt MQDestination , zostanie przyjęta niezarządzana subskrypcja. Należy upewnić się, że ustawione opcje subskrypcji są spójne z zarządzaną lub niezarządzaną subskrypcją.

miejsce docelowe
destination jest instancją MQQueue . Podanie wartości *destination*powoduje, że program MQTopic jest otwierany jako niezarządzana subskrypcja. Publikacje w tym temacie są dostarczane do kolejki, do której dostęp jest uzyskiwany jako *destination*.

topicName
Łańcuch tematu, który jest drugą częścią nazwy tematu. Łańcuch *topicName* jest konkatenowany z łańcuchem tematu zdefiniowanym w obiekcie tematu administracyjnego produktu *topicObject* . Parametr *topicName* można ustawić na wartość NULL. W takim przypadku nazwa tematu jest definiowana przez łańcuch tematu w pliku *topicObject*.

topicObject
Na wejściu *topicObject* jest nazwą obiektu tematu zawierającego łańcuch tematu, który stanowi pierwszą część nazwy tematu. Łańcuch tematu w pliku *topicObject* jest konkatenowany z łańcuchem *topicName*. Reguły tworzenia łańcuchów tematów są zdefiniowane w sekcji Łączenie łańcuchów tematów.

W danych wyjściowych *topicObject* zawiera nazwę obiektu tematu administracyjnego, który jest najbardziej zgodny w drzewie tematów z tematem identyfikowanym przez łańcuch tematu.

openAs
Dostęp do tematu w celu publikowania lub subskrybowania. Parametr może zawierać tylko jedną z następujących opcji:

- MQC.MQTOPIC_OPEN_AS_SUBSCRIPTION
- MQC.MQTOPIC_OPEN_AS_PUBLICATION

opcje

Połącz opcje sterujące otwieraniem tematu dla publikacji lub subskrypcji. Stałe MQC.MQSO_* umożliwiają dostęp do tematu dla subskrypcji, a stałe MQC.MQOO_* umożliwiają dostęp do tematu dla publikacji.

Jeśli wymagana jest więcej niż jedna opcja, dodaj wartości lub połącz wartości opcji przy użyciu operatora bitowej OR .

Identyfikator alternateUser
Podaj alternatywny ID użytkownika, który jest używany do sprawdzania wymaganych uprawnień do zakończenia operacji. Jeśli w parametrze opcji ustawiono wartość MQC.MQOO_ALTERNATE_USER_AUTHORITY lub MQC.MQSO_ALTERNATE_USER_AUTHORITY , należy podać wartość *alternateUserId*.

subscriptionName
Parametr *subscriptionName* jest wymagany, jeśli podano opcje MQC.MQSO_DURABLE lub MQC.MQSO_ALTER . W obu przypadkach program MQTopic jest niejawnie otwierany do subskrypcji. Wyjątek jest zgłaszany, jeśli właściwość MQC.MQSO_DURABLE jest ustawiona, a subskrypcja istnieje lub jeśli właściwość MQC.MQSO_ALTER jest ustawiona, a subskrypcja nie istnieje.

Właściwości
Ustaw dowolną ze specjalnych właściwości subskrypcji wymienionych przy użyciu tabeli mieszającej. Podane pozycje w tabeli mieszającej są aktualizowane wartościami wyjściowymi. Pozycje nie są dodawane do tabeli mieszającej w celu raportowania wartości wyjściowych.

- 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

Interfejs IMQObjectTrigger.NET

Zaimplementuj interfejs IMQObjectTrigger , aby przetwarzać komunikaty przekazywane przez monitor **runmqdnm**.NET .

Interfejs

```
public interface IBM.WMQMonitor.IMQObjectTrigger();
```

W zależności od tego, czy w komendzie **runmqdnm** określono element sterujący punktu synchronizacji, komunikat jest usuwany z kolejki przed lub po powrocie z metody Execute .

metody

void Execute (MQQueueManager *queueManager*, MQQueue *queue*, MQMessage *message*, string *param*);

queueManager
Menedżer kolejek udostępniający monitorowaną kolejkę.

kolejka
Monitorowana kolejka.

komunikat
Komunikat odczytany z kolejki.

Param
Dane przekazane z UserParameter.

Interfejs MQC.NET

Aby odwołać się do stałej MQI , należy poprzedzić nazwę stałej przedrostkiem MQC.. MQC definiuje wszystkie stałe używane przez MQI.

Interfejs

```
System.Object
        |
        └─ IBM.WMQ.MQC
```

```
public interface IBM.WMQ.MQC extends System.Object;
```

Przykład

```
MQQueue queue;
queue.closeOptions = MQC.MQCO_DELETE;
```

Identyfikatory zestawów znaków dla aplikacji .NET

Opisy zestawów znaków, które można wybrać w celu zakodowania komunikatów produktu .NET IBM MQ

Zestaw znaków	Opis
37	ibm037
437	ibm437 /PC Original
500	ibm500
819	iso-8859-1 / latin1 / ibm819

Zestaw znaków	Opis
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 grecki
775	ibm775 /PC bałtycki
813	iso-8859-7 /grecki/ ibm813
838	ibm838
850	ibm850 /PC Latin 1
852	ibm852 /PC Latin 2
855	ibm855 /PC Cyrillic
856	ibm856
857	ibm857 /PC turecki
860	ibm860 /PC portugalski
861	ibm861 /PC islandzki
862	ibm862 /PC hebrajski
863	ibm863 /PC francuski (Kanada)
864	ibm864 /PC-arabski
865	ibm865 /PC Nordic
866	ibm866 /PC rosyjski
868	ibm868
869	ibm869 /PC Modern Greek
870	ibm870
871	ibm871
874	ibm874
875	ibm875
912	iso-8859-2 / latin2 / ibm912
913	iso-8859-3 / latin3 / ibm913

Zestaw znaków	Opis
914	iso-8859-4 / latin4 / ibm914
915	iso-8859-5 /cyrylica/ ibm915
916	iso-8859-8 /hebrew/ ibm916
918	ibm918
920	iso-8859-9 / latin5 / ibm920
921	ibm921
922	ibm922
930	ibm930
932	PC japoński
933	ibm933
935	ibm935
937	ibm937
939	ibm939
942	ibm942
943	ibm943
948	ibm948
949	ibm949
950	ibm950 /Big 5 chiński tradycyjny
954	EUCJIS
964	ibm964 /CNS 11643 chiński tradycyjny
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 Latin 2
1251	Windows cyrylica
1252	Windows Latin 1
1253	Windows grecki

Zestaw znaków	Opis
1254	Windows turecki
1255	Windows hebrajski
1256	Windows arabski
1257	Windows bałtycki
1258	Windows wietnamski
1381	ibm1381
1383	ibm1383
2022	JIS
5601	ksc-5601 koreański
33722	ibm33722

Klasy IBM MQ C++

Klasy języka C++ IBM MQ hermetyzują interfejs MQI (Message Queue Interface) systemu IBM MQ . Istnieje pojedynczy plik nagłówkowy C + +, **imqi.hpp**, który obejmuje wszystkie te klasy.

Dla każdej klasy wyświetlane są następujące informacje:

Diagram hierarchii klas
Diagram klas przedstawiający klasę w relacji dziedziczenia z jej bezpośrednimi klasami nadrzędnymi (jeśli istnieją).

Inne istotne klasy
Odsyłacze dokumentów do innych odpowiednich klas, takich jak klasy macierzyste, oraz klas obiektów używanych w podpisach metod.

Atrybuty obiektu
Atrybuty klasy. Są to atrybuty dodatkowe w stosunku do atrybutów zdefiniowanych dla klas macierzystych. Wiele atrybutów odzwierciedla elementy struktury danych IBM MQ (patrz “Odwołanie do interfejsu MQI i C++” na stronie 1840). Szczegółowe opisy zawiera sekcja “Atrybuty obiektów” na stronie 824.

Konstruktory
Sygnatury metod specjalnych używanych do tworzenia obiektu klasy.

Metody obiektów (publiczne)
Sygnatury metod, które wymagają instancji klasy dla swojej operacji i które nie mają ograniczeń użycia.

Tam, gdzie ma to zastosowanie, wyświetlane są również następujące informacje:

Metody klasy (publiczne)
Sygnatury metod, które nie wymagają instancji klasy dla swojej operacji i które nie mają ograniczeń użycia.

Metody przeciążone (klasa nadrzędna)
Sygnatury tych metod wirtualnych, które są zdefiniowane w klasach macierzystych, ale wykazują różne, polimorficzne zachowanie dla tej klasy.

Metody obiektów (zabezpieczone)
Sygnatury metod, które wymagają instancji klasy dla swojej operacji i są zarezerwowane do użycia przez implementacje klas pochodnych. Ta sekcja jest interesująca tylko dla programów piszących klasy, a nie dla użytkowników klas.

Dane obiektu (zabezpieczone)
Szczegóły implementacji dla danych instancji obiektu dostępne dla implementacji klas pochodnych. Ta sekcja jest interesująca tylko dla programów piszących klasy, a nie dla użytkowników klas.

Kody przyczyny
Wartości MQRC_ * (patrz sekcja Kody zakończenia i przyczyny funkcji API), których można oczekiwać od metod, które nie powiodły się. Pełna lista kodów przyczyny, które mogą wystąpić dla obiektu klasy, znajduje się w dokumentacji klasy nadrzędnej. Udokumentowana lista kodów przyczyny dla klasy nie zawiera kodów przyczyny dla klas nadrzędnych.

Uwaga:

1. Obiekty tych klas nie są wątkowo bezpieczne. Zapewnia to optymalną wydajność, ale należy uważać, aby nie uzyskać dostępu do żadnego obiektu z więcej niż jednego wątku.
2. W przypadku programu wielowątkowego zaleca się użycie osobnego obiektu menedżera ImqQueuedla każdego wątku. Każdy obiekt menedżera musi mieć własną niezależną kolekcję innych obiektów, zapewniając izolowanie obiektów w różnych wątkach.

Są to następujące klasy:

- "Klasa ImqAuthenticationRecord języka C++" na stronie 1857
- "Klasa C++ ImqBinary" na stronie 1860
- "Klasa ImqCache C++" na stronie 1862
- "Klasa ImqChannel C++" na stronie 1864
- "ImqCICSBridgeHeader (klasa C + +)" na stronie 1870
- "ImqDeadLetterHeader klasa C + +" na stronie 1876
- "Klasa ImqDistributionList C++" na stronie 1879
- "Klasa ImqError C++" na stronie 1880
- "Klasa C++ ImqGetMessageOptions" na stronie 1881
- "Klasa C++ ImqHeader" na stronie 1885
- "ImqIMSBridgeHeader (klasa C + +)" na stronie 1886
- "Klasa ImqItem C++" na stronie 1889
- "Klasa ImqMessage C++" na stronie 1891
- "Klasa ImqMessageTracker C++" na stronie 1898
- "Klasa ImqNamelist C++" na stronie 1901
- "Klasa C + + ImqObject" na stronie 1902
- "Klasa ImqProcess C++" na stronie 1908
- "ImqPutMessageOptions klasa C++" na stronie 1910
- "Klasa ImqQueue C++" na stronie 1912
- "Klasa ImqQueueManager C++" na stronie 1923
- "Klasa C++ nagłówka ImqReference" na stronie 1940
- "Klasa ImqString C++" na stronie 1942
- "Klasa ImqTrigger C++" na stronie 1947
- "Klasa C++ nagłówka ImqWork" na stronie 1950

Odwołanie do interfejsu MQI i C++

Ta kolekcja tematów zawiera informacje dotyczące interfejsu MQI w języku C + +.

Należy przeczytać te informacje razem z informacjami zawartymi w sekcji "Typy danych używane w MQI" na stronie 236.

Ta tabela odnosi struktury danych MQI do klas C++ i plików włączanych. W poniższych tematach przedstawiono informacje uzupełniające dla każdej klasy C + +. Te odwołania odnoszą się do korzystania z bazowych interfejsów proceduralnych IBM MQ . Klasy ImqBinary, ImqDistributionList i ImqString nie mają atrybutów należących do tej kategorii i są wykluczone.

Tabela 845. Odwołanie do struktury danych, klasy i pliku włączanego		
Struktura danych	**Klasa**	**Plik włączkowy**
MQAIR	Rekord ImqAuthentication	imqair.hpp
	ImqBinary	imqbin.hpp
	ImqCache	imqcac.hpp
MQCD	ImqChannel	imqchl.hpp
MQCIH,	ImqCICSBridgeHeader	imqcih.hpp
MQDLH	ImqDeadLetterHeader	imqdlh.hpp
ZMQOR	Lista ImqDistribution	imqdst.hpp
	ImqError	imqerr.hpp
MQGMO	ImqGetMessageOptions	imqgmo.hpp
	ImqHeader	imqhdr.hpp
MQIIH.	ImqIMSBridgeHeader	imqiih.hpp
	ImqItem	imqitm.hpp
MQMD	ImqMessage	imqmsg.hpp
	Śledzenie 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	Menedżer ImqQueue	imqmgr.hpp
MQRMH	Nagłówek ImqReference	imqrfh.hpp
	ImqString	imqstr.hpp
MQTM	ImqTrigger	imqtrg.hpp
MQTMC		
MQTMC2	ImqTrigger	imqtrg.hpp
MQXQH		
MQWIH	Nagłówek ImqWork	imqwih.hpp

Odwołanie do rekordu ImqAuthentication

Odniesienie między atrybutami, strukturami danych, polami i wywołaniami klasy ImqAuthenticationRecord języka C + +.

Tabela 846. Atrybuty, struktury danych, pola i wywołania			
Atrybut	**Struktura danych**	**Pole**	**Zadzwoń**
Typ połączenia	MQAIR	AuthInfoConnName	MQCONNX

Tabela 846. Atrybuty, struktury danych, pola i wywołania (kontynuacja)			
Atrybut	**Struktura danych**	**Pole**	**Zadzwoń**
Hasło	MQAIR	Hasło_LDAP	MQCONNX
typ	MQAIR	AuthInfoTyp	MQCONNX
nazwa użytkownika,	MQAIR	LDAPUserNamePtr	MQCONNX
	MQAIR	Przesunięcie LDAPUserName	MQCONNX
	MQAIR	Długość LDAPUserName	MQCONNX

Odwołanie ImqCache

Odwołanie do atrybutów i wywołań klasy ImqCache w języku C + +.

Tabela 847. Atrybuty i wywołania	
Atrybut	**Zadzwoń**
bufor automatyczny	MQGET
długość buforu	MQGET
wskaźnik buforu	MQGET, MQPUT
Długość danych	MQGET
Pozycja danych	MQGET
wskaźnik danych	MQGET
długość komunikatu	MQGET, MQPUT

Odwołanie ImqChannel

Odwołanie do atrybutów, struktur danych, pól i wywołań klasy ImqChannel w języku C + +.

Tabela 848. Atrybuty, struktury danych, pola i wywołania			
Atrybut	**Struktura danych**	**Pole**	**Zadzwoń**
bicie serca	MQCD	BatchHeartbeat	MQCONNX
nazwa kanału	MQCD	ChannelName	MQCONNX
Typ połączenia	MQCD	ConnectionName	MQCONNX
	MQCD	ShortConnectionNazwa	MQCONNX
Kompresja nagłówka	MQCD	Lista HdrComp	MQCONNX
interwał pulsu	MQCD	HeartbeatInterval	MQCONNX
Interwał sprawdzania połączenia	MQCD	Przedział czasu KeepAlive	MQCONNX
Adres lokalny	MQCD	LocalAddress	MQCONNX
Maksymalna długość komunikatu	MQCD	MaxMsg	MQCONNX
Kompresja komunikatu	MQCD	Lista MsgComp	MQCONNX
Nazwa trybu	MQCD	ModeName	MQCONNX
Hasło	MQCD	Hasło	MQCONNX

Tabela 848. Atrybuty, struktury danych, pola i wywołania (kontynuacja)			
Atrybut	**Struktura danych**	**Pole**	**Zadzwoń**
liczba wyjść odbierania	MQCD		MQCONNX
nazwy wyjść odbierania	MQCD	ReceiveExit	MQCONNX
	MQCD	ReceiveExitsZdefiniowane	MQCONNX
	MQCD	ReceiveExitPtr	MQCONNX
odbierz dane użytkownika	MQCD	Dane ReceiveUser	MQCONNX
	MQCD	ReceiveUserDataPtr	MQCONNX
Nazwa wyjścia zabezpieczeń	MQCD	SecurityExit	MQCONNX
dane użytkownika dotyczące bezpieczeństwa	MQCD	Dane SecurityUser	MQCONNX
liczba wyjść wysyłania	MQCD		MQCONNX
nazwy wyjść wysyłania	MQCD	SendExit	MQCONNX
	MQCD	SendExits-zdefiniowane	MQCONNX
	MQCD	SendExitPtr	MQCONNX
wyślij dane użytkownika	MQCD	Dane SendUser	MQCONNX
	MQCD	SendUserDataPtr	MQCONNX
CipherSpec SSL	MQCD	Specyfikacja sslCipher	MQCONNX
Typ uwierzytelniania klienta SSL	MQCD	Uwierzytelnianie sslClient	MQCONNX
Nazwa węzła sieci SSL	MQCD	SSLPEERNAME	MQCONNX
Nazwa programu transakcyjnego	MQCD	TpName	MQCONNX
Typ transportu	MQCD	TransportType	MQCONNX
ID użytkownika	MQCD	UserIdentifier	MQCONNX

Odwołanie ImqCICSBridgeHeader

Odwołanie do atrybutów, struktur danych i pól dla klasy ImqCICSBridgeHeader języka C + +.

Tabela 849. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
kod nieprawidłowego zakończenia mostu	MQCIH,	AbendCode
Deskryptor ADS	MQCIH,	AdsDescriptor
identyfikator uwagi	MQCIH,	AttentionId
element uwierzytelniający	MQCIH,	Authenticator
kod zakończenia mostu	MQCIH,	Kod BridgeCompletion
przesunięcie błędu mostu	MQCIH,	ErrorOffset
kod przyczyny mostu	MQCIH,	BridgeReason
kod anulowania mostu	MQCIH,	CancelCode
zadanie konwersacyjne	MQCIH,	ConversationalTask

Tabela 849. Odwzorowanie atrybutów, struktur danych i pól (kontynuacja)		
Atrybut	**Struktura danych**	**Pole**
pozycja kursora	MQCIH,	CursorPosition
token narzędzia	MQCIH,	Udogodnienia
czas przechowywania obiektu	MQCIH,	FacilityKeepCzas
narzędzie podobne do	MQCIH,	FacilityLike
function (funkcja)	MQCIH,	Funkcja
przedział czasu oczekiwania na pobranie	MQCIH,	Przedział czasu GetWait
Typ odsyłacza	MQCIH,	LinkType
identyfikator następnej transakcji	MQCIH,	Identyfikator NextTransaction
długość danych wyjściowych	MQCIH,	OutputDataDługość
format odpowiedzi	MQCIH,	Format ReplyTo
kod powrotu mostu	MQCIH,	ReturnCode
kod początkowy	MQCIH,	StartCode
status zakończenia zadania	MQCIH,	Status TaskEnd
Identyfikator transakcji	MQCIH,	TransactionId
Kontrola uow	MQCIH,	UowControl
wersja	MQCIH,	Wersja

Odwołanie ImqDeadLetterHeader

Odwołanie do atrybutów, struktur danych i pól dla klasy języka C + + ImqDeadLetterHeader .

Tabela 850. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
kod przyczyny niedostarczonych komunikatów	MQDLH	Przyczyna
Nazwa menedżera kolejek docelowych	MQDLH	Docelowy_menedżer_kolejek
nazwa kolejki docelowej	MQDLH	DestQName
Nazwa aplikacji umieszczającej	MQDLH	Nazwa_aplikacji_wstawiającej
Typ aplikacji umieszczającej	MQDLH	Typ_aplikacji_wstawiającej
Data umieszczenia	MQDLH	PutDate
Czas umieszczenia	MQDLH	PutTime

Odwołanie ImqError

Odwołanie do atrybutów i wywołań klasy ImqError C + +.

Tabela 851. Atrybuty i wywołania	
Atrybut	**Zadzwoń**
kod zakończenia	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

Tabela 851. Atrybuty i wywołania (kontynuacja)	
Atrybut	**Zadzwoń**
reason code (kod przyczyny)	MQBACK, MQBEGIN, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQDISC, MQGET, MQINQ, MQOPEN, MQPUT, MQSET

ImqGetMessageOptions odwołanie

Odwołanie do atrybutów, struktur danych i pól dla klasy C++ ImqGetMessageOptions .

Tabela 852. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
Status grupy	MQGMO	GroupStatus
opcje dopasowywania	MQGMO	MatchOptions
znacznik komunikatu	MQGMO	MessageToken
opcje	MQGMO	Opcje
przetłumaczona nazwa kolejki	MQGMO	ResolvedQName
zwrócona długość	MQGMO	ReturnedLength
segmentacja	MQGMO	Segmentacja
status segmentu	MQGMO	SegmentStatus
	MQGMO	Signal1
	MQGMO	Signal2
Uczestnictwo w punkcie synchronizacji	MQGMO	Opcje
Interwał oczekiwania	MQGMO	WaitInterval

Odwołanie ImqHeader

Odwołanie do atrybutów, struktur danych i pól dla klasy języka C + + ImqHeader .

Tabela 853. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
zestaw znaków	MQDLH, MQIIH	CodedCharSetId
kodowanie	MQDLH, MQIIH	Kodowanie
format	MQDLH, MQIIH	Format
flagi nagłówka	MQIIH, MQRMH	Flagi

Odwołanie ImqIMSBridgeHeader

Odwołanie do atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord w języku C + +.

Tabela 854. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
element uwierzytelniający	MQIIH.	Authenticator
tryb zatwierdzania	MQIIH.	CommitMode
przesłonięcie terminalu logicznego	MQIIH.	LTermOverride

Tabela 854. Odwzorowanie atrybutów, struktur danych i pól (kontynuacja)		
Atrybut	**Struktura danych**	**Pole**
nazwa odwzorowania usług formatu komunikatu	MQIIH.	MFSMapName
format odpowiedzi	MQIIH.	Format ReplyTo
zasięg zabezpieczeń	MQIIH.	SecurityScope
identyfikator instancji transakcji	MQIIH.	Identyfikator TranInstance
Stan transakcji	MQIIH.	TranState

Odwołanie ImqItem

Odwołanie do atrybutów i wywołań klasy języka C + + ImqItem .

Tabela 855. Atrybuty i wywołania	
Atrybut	**Zadzwoń**
identyfikator struktury	MQGET

Odwołanie ImqMessage

Odwołanie do atrybutów, struktur danych, pól i wywołań klasy języka C + + ImqMessage .

Tabela 856. Atrybuty, struktury danych, pola i wywołania			
Atrybut	**Struktura danych**	**Pole**	**Zadzwoń**
Dane identyfikatora aplikacji	MQMD	Dane_tożsamości_aplikacji	
Dane źródłowe aplikacji	MQMD	Dane_pochodzenia_aplikacji	
Liczba wycofań	MQMD	BackoutCount	
zestaw znaków	MQMD	CodedCharSetId	
kodowanie	MQMD	Kodowanie	
Koniec ważności	MQMD	Utrata ważności	
format	MQMD	Format	
Flagi komunikatów	MQMD	MsgFlags	
typ komunikatu	MQMD	MsgType	
Przesunięcie	MQMD	Depozycja	
Oryginalna długość	MQMD	OriginalLength	
trwałość	MQMD	Trwałość	
priorytet	MQMD	Priorytet	
Nazwa aplikacji umieszczającej	MQMD	Nazwa_aplikacji_wstawiającej	
Typ aplikacji umieszczającej	MQMD	Typ_aplikacji_wstawiającej	
Data umieszczenia	MQMD	PutDate	
Czas umieszczenia	MQMD	PutTime	

Tabela 856. Atrybuty, struktury danych, pola i wywołania (kontynuacja)			
Atrybut	**Struktura danych**	**Pole**	**Zadzwoń**
nazwa menedżera kolejek odpowiedzi	MQMD	ReplyToQMgr	
nazwa kolejki odpowiedzi	MQMD	ReplyToQ	
raportowanie	MQMD	Raport	
numer kolejny	MQMD	Numer_kolejny_komunikatu	
łączna długość komunikatu		DataLength	MQGET
ID użytkownika	MQMD	UserIdentifier	

ImqMessageOdwołanie do śledzenia

Odwołanie do atrybutów, struktur danych i pól dla klasy C++ narzędzia śledzącego ImqMessage.

Tabela 857. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
Token rozliczania	MQMD	AccountingToken
Identyfikator korelacji	MQMD	CorrelId
Opinia	MQMD	Opinie
Identyfikator grupy	MQMD	GroupId
Identyfikator wiadomości	MQMD	MsgId

Odwołanie ImqNamelist

Odwołanie do atrybutów, zapytań i wywołań dla klasy ImqNamelist języka C + +.

Tabela 858. Atrybuty, zapytania i wywołania		
Atrybut	**Zapytanie**	**Zadzwoń**
Liczba nazw	MQIA_NAME_COUNT (liczba nazw MQIA)	MQINQ
Nazwa listy nazw	Nazwa listy MQCA_NAMELIST_NAME	MQINQ

Odwołanie ImqObject

Odwołanie do atrybutów, struktur danych, pól, zapytań i wywołań klasy C++ ImqObject .

Tabela 859. Atrybuty, struktury danych, pola, zapytania i wywołania				
Atrybut	**Struktura danych**	**Pole**	**Zapytanie**	**Zadzwoń**
Data zmiany			MQCA_ALTERATION_DATE	MQINQ
Godzina zmiany			MQCA_ALTERATION_TIME	MQINQ
Alternatywny identyfikator użytkownika	ZMQOD	Identyfikator AlternateUser		
Alternatywny identyfikator zabezpieczeń				

Tabela 859. Atrybuty, struktury danych, pola, zapytania i wywołania (kontynuacja)				
Atrybut	**Struktura danych**	**Pole**	**Zapytanie**	**Zadzwoń**
opcje zamykania				MQCLOSE
opis			MQCA_Q_DESC, MQCA_Q_MGR_DESC, MQCA_PROCESS_DESC	MQINQ
nazwa	ZMQOD	ObjectName	MQCA_Q_MGR_NAME, MQCQ_Q_NAME, MQCA_PROCESS_NAME	MQINQ
Opcje otwarcia				MQOPEN
status otwarcia				MQOPEN, MQCLOSE
identyfikator menedżera kolejek	identyfikator menedżera kolejek		MQCA_Q_MGR_IDENTIFIER	MQINQ

Odwołanie ImqProcess

Odwołanie do atrybutów, zapytań i wywołań klasy ImqAuthenticationRecord w języku C + +.

Tabela 860. Atrybuty, zapytania i wywołania		
Atrybut	**Zapytanie**	**Zadzwoń**
Identyfikator aplikacji	MQCA_APPL_ID (Identyfikator aplikacji MQ)	MQINQ
Typ aplikacji	TYP_APLIKACJI_MQIA	MQINQ
Dane środowiska	MQCA_ENV_DATA	MQINQ
Dane użytkownika	MQCA_USER_DATA	MQINQ

ImqPutMessageOptions odwołanie

Odwołanie do atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord w języku C + +.

Tabela 861. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
odwołanie do kontekstu	MQPMO	Kontekst
	MQPMO	Liczba: InvalidDest
	MQPMO	Liczba: KnownDest
opcje	MQPMO	Opcje
pola rekordu	MQPMO	PutMsgRecFields
rozstrzygnięta nazwa menedżera kolejek	MQPMO	ResolvedQMgrNazwa
przetłumaczona nazwa kolejki	MQPMO	ResolvedQName
	MQPMO	Limit czasu
	MQPMO	Liczba: UnknownDest

Tabela 861. Odwzorowanie atrybutów, struktur danych i pól (kontynuacja)		
Atrybut	**Struktura danych**	**Pole**
Uczestnictwo w punkcie synchronizacji	MQPMO	Opcje

Odwołanie ImqQueue

Odwołanie do atrybutów, struktur danych, pól, zapytań i wywołań klasy C++ ImqQueue .

Tabela 862. Odwołanie ImqQueue				
Atrybut	**Struktura danych**	**Pole**	**Zapytanie**	**Zadzwoń**
nazwa kolejki wycofanych komunikatów			MQCA_BACKOUT_REQ_Q_NAME	MQINQ
Próg wycofania			MQIA_BACKOUT_THRESHOLD (próg wycofania MQ)	MQINQ
nazwa kolejki podstawowej			MQCA_BASE_Q_NAME	MQINQ
nazwa klastra			MQCA_NAZWA_KLASTRA	MQINQ
Nazwa listy nazw klastrów			MQCA_CLUSTER_NAMELIST (lista nazw klastrów MQ)	MQINQ
Klasyfikacja obciążenia klastrów			MQIA_CLWL_Q_RANK	MQINQ
Priorytet obciążenia klastrów			MQIA_CLWL_Q_PRIORITY,	MQINQ
Kolejka użycia obciążenia klastra			MQIA_CLWL_USEQ	MQINQ
data utworzenia			MQCA_CREATION_DATE (data utworzenia MQCA)	MQINQ
Czas utworzenia			MQCA_CREATION_TIME (czas utworzenia MQCA)	MQINQ
Bieżące zapełnienie			MQIA_BIEŻĄCE_ZAPEŁNIENIE_KOLEJKI	MQINQ
powiązanie domyślne			MQIA_DEF_BIND	MQINQ
Domyślna opcja otwarcia wejścia			MQIA_DEF_INPUT_OPEN_OPTION	MQINQ
Trwałość domyślna			MQIA_DEF_PERSISTENCE	MQINQ
Domyślny priorytet			MQIA_DEF_PRIORITY,	MQINQ
Typ definicji			TYP_definicji_MQIA	MQINQ
Zdarzenie o dużej głębokości			MQIA_Q_DEPTH_HIGH_EVENT	MQINQ
górny limit głębokości			MQIA_Q_DEPTH_HIGH_LIMIT	MQINQ
zdarzenie niskiego poziomu głębokości			MQIA_Q_DEPTH_LOW_EVENT	MQINQ

Tabela 862. Odwołanie ImqQueue (kontynuacja)

Atrybut	Struktura danych	Pole	Zapytanie	Zadzwoń
dolny limit głębokości			MQIA_Q_DEPTH_LOW_LIMIT	MQINQ
maksymalna głębokość zdarzenia			MQIA_Q_DEPTH_MAX_LIMIT	MQINQ
listy dystrybucyjne			MQIA_DIST_LISTS	MQINQ, MQSET
nazwa kolejki dynamicznej	ZMQOD	DynamicQName		
Zapisane wycofane komunikaty			MQIA_HARDEN_GET_BACKOUT	MQINQ
Typ indeksu			MQIA_INDEX_TYPE	MQINQ
Nie zezwalaj na pobieranie			MQIA_INHIBIT_GET	MQINQ, MQSET
Zablokuj umieszczenie			MQIA_INHIBIT_PUT	MQINQ, MQSET
Nazwa kolejki inicjacji			MQCA_INITIATION_Q_NAME (nazwa kolejki inicjowania MQCA)	MQINQ
Maksymalna głębokość			MQIA_MAX_Q_DEPTH	MQINQ
Maksymalna długość komunikatu			MQIA_MAX_MSG_LENGTH	MQINQ
Kolejność dostarczania komunikatów			MQIA_MSG_DELIVERY_SEQUENCE (sekwencja dostarczania komunikatów)	MQINQ
następna kolejka rozproszona				
Klasa komunikatów nietrwałych			KLASA MQIA_NPM_CLASS	MQINQ
Liczba otwartych wejść			MQIA_OPEN_INPUT_COUNT	MQINQ
Liczba otwartych wyjść			MQIA_OPEN_OUTPUT_COUNT	MQINQ
poprzednia kolejka rozproszona				
Nazwa procesu			NAZWA_PROCESU_MQCA_MQCA_NAME	MQINQ
Rozliczanie kolejek			LICZNIK MQIA_ACTING_Q	MQINQ
Nazwa menedżera kolejek	ZMQOD	ObjectQMgrNazwa		
Monitorowanie kolejek			MQIA_MONITORING_Q (kolejka MQIA)	MQINQ
Statystyka kolejek			MQIA_STATISTICS_Q (kolejka MQIA)	MQINQ

Tabela 862. Odwołanie ImqQueue (kontynuacja)				
Atrybut	**Struktura danych**	**Pole**	**Zapytanie**	**Zadzwoń**
Typ kolejki			MQIA_Q_TYPE	MQINQ
Nazwa zdalnego menedżera kolejek			MQCA_REMOTE_Q_MGR_NAME	MQINQ
Nazwa zdalnej kolejki			MQCA_REMOTE_Q_NAME	MQINQ
rozstrzygnięta nazwa menedżera kolejek	ZMQOD	ResolvedQMgrNazwa		
przetłumaczona nazwa kolejki	ZMQOD	ResolvedQName		
Interwał przechowywania			INTERWAŁ MQIA_RETENTION_INTERVAL	MQINQ
zasięg			ZAKRES MQIA_SCOPE	MQINQ
interwał usług			MQIA_Q_SERVICE_INTERVAL	MQINQ
zdarzenie interwału usług			MQIA_Q_SERVICE_INTERVAL_EVENT	MQINQ
Możliwość współużytkowania			MOŻLIWOŚĆ współużytkowania mqia_shareability	MQINQ
klasa pamięci masowej			KLASA MQCA_STORAGE_CLASS	MQINQ
Nazwa kolejki transmisji			MQCA_XMIT_Q_NAME	MQINQ
Kontrola wyzwalacza			MQIA_TRIGGER_CONTROL	MQINQ, MQSET
Dane wyzwalacza			MQCA_TRIGGER_DATA	MQINQ, MQSET
Wyzwalacz uruchamiany zapełnieniem			MQIA_TRIGGER_DEPTH	MQINQ, MQSET
Priorytet komunikatu wyzwalacza			MQIA_TRIGGER_MSG_PRIORITY,	MQINQ, MQSET
typ wyzwalacza			MQIA_TRIGGER_TYPE	MQINQ, MQSET
wykorzystanie			UŻYCIE MQIA	MQINQ

Odwołanie do menedżera ImqQueue

Odwołanie do atrybutów, struktur danych, pól, zapytań i wywołań dla klasy C++ programu ImqQueueManager.

Tabela 863. Atrybuty, struktury danych, pola, zapytania i wywołania

Atrybut	Struktura danych	Pole	Zapytanie	Zadzwoń
przesłonięcie połączeń rozliczeniowych			MQIA_ACCOUNTING_CONN_OVERRIDE (mqia_accounting_override)	MQINQ
Interwał rozliczania			MQIA_ACCOUNTING_INTERVAL	MQINQ
Zapis aktywności			DZIAŁANIE MQIA_ACTIVITY_RECORDING	MQINQ
Sprawdzenie dołączenia nowego MCA			MQIA_XX_ENCODE_CASE_ONE sprawdzenie_adoptowania TNEWMCA	MQINQ
Typ dołączenia nowego MCA			MQIA_XX_ENCODE_CASE_ONE typ_adopcji TNEWMCA	MQINQ
Typ uwierzytelniania	Protokół MQCSP	AuthenticationType		MQCONNX
zdarzenie uprawnień			MQIA_AUTHORITY_EVENT	MQINQ
opcje początku	Obiekt MQBO	Opcje		MQBEGIN
zdarzenie mostu			ZDARZENIEM MQIA_BRIDGE_EVENT	MQINQ
Automatyczna definicja kanału			MQIA_CHANNEL_AUTO_DEF	MQINQ
zdarzenie automatycznej definicji kanału			MQIA_CHANNEL_AUTO_EVENT	Obszar MQIA
Wyjście automatycznej definicji kanału			MQIA_CHANNEL_AUTO_EXIT	Obszar MQIA
zdarzenie kanału			ZDARZENIE MQIA_CHANNEL_EVENT	MQINQ
Adaptery inicjatora kanału			MQIA_CHINIT_ADAPTERS	MQINQ
Kontrola inicjatora kanału			MQIA_CHINIT_CONTROL,	MQINQ
Programy rozsyłające inicjatora kanału			MQIA_CHINIT_DISPATCHERS (programy rozsyłające)	MQINQ
Autostart śledzenia inicjatora kanału			MQIA_CHINIT_TRACE_AUTO_START	MQINQ
Wielkość tabeli śledzenia inicjatora kanału			MQIA_CHINIT_TRACE_TABLE_SIZE	MQINQ
Monitorowanie kanałów			MQIA_MONITORING_CHANNEL,	MQINQ

Tabela 863. Atrybuty, struktury danych, pola, zapytania i wywołania (kontynuacja)

Atrybut	Struktura danych	Pole	Zapytanie	Zadzwoń
odwołanie do kanału	MQCD	ChannelType		MQCONNX
Statystyka kanałów			MQIA_STATISTICS_CHANNEL,	MQINQ
zestaw znaków			MQIA_CODED_CHAR_SET_ID	MQINQ
Monitorowanie nadawcy klastrów			Usługa MQIA_MONITORING_AUTO_CLUSSDR	MQINQ
Statystyka nadawcy klastrów			MQIA_STATISTICS_AUTO_CLUSSDR	MQINQ
Dane obciążenia klastra			MQCA_CLUSTER_WORKLOAD_DATA (dane obciążenia klastra MQCA)	MQINQ
Wyjście obciążenia klastra			MQCA_CLUSTER_WORKLOAD_EXIT,	MQINQ
Długość obciążenia klastra			MQIA_CLUSTER_WORKLOAD_LENGTH (długość obciążenia klastra MQIA)	MQINQ
mru obciążenia klastra			MQIA_CLWL_MRU_CHANNELS	MQINQ
Kolejka użycia obciążenia klastra			MQIA_CLWL_USEQ	MQINQ
zdarzenie komendy			MQIA_COMMAND_EVENT	MQINQ
nazwa kolejki wejściowej komendy			MQCA_COMMAND_INPUT_Q_NAME	MQINQ
poziom komendy			POZIOM_KOMENDY MQIA_COMMAND_LEVEL	MQINQ
Kontrola serwera komend			MQIA_CMD_SERVER_CONTROL	MQINQ
Opcje połączenia	MQCNO	Opcje		MQCONN, MQCONNX
Identyfikator połączenia	MQCNO	ConnectionId		MQCONNX
status połączenia				MQCONN, MQCONNX, MQDISC
Znacznik połączenia	MQCD	ConnTag		MQCONNX
Sprzęt szyfrujący	MQSCO	CryptoHardware		MQCONNX
nazwa kolejki niedostarczonych komunikatów			MQCA_DEAD_LETTER_Q_NAME	MQINQ

Tabela 863. Atrybuty, struktury danych, pola, zapytania i wywołania (kontynuacja)

Atrybut	Struktura danych	Pole	Zapytanie	Zadzwoń
domyślna nazwa kolejki transmisji			MQCA_DEF_XMIT_Q_NAME	MQINQ
listy dystrybucyjne			MQIA_DIST_LISTS	MQINQ
grupa dns			GRUPA_DN_MQCA	MQINQ
dns wlm			WLM MQIA_DNS	MQINQ
pierwszy rekord uwierzytelniania	MQSCO	AuthInfoRecOffset		MQCONNX
	MQSCO	AuthInfoRecPtr		MQCONNX
Zdarzenie wstrzymania			MQIA_INHIBIT_EVENT	MQINQ
Wersja adresu IP			WERSJA_ADRESU_IP_MQIA	MQINQ
repozytorium kluczy	MQSCO	KeyRepository		MQCONNX
liczba resetów klucza	MQSCO	KeyResetLiczba		MQCONNX
Zegar nasłuchiwania			LICZNIK MQIA_LISTENER_TIMER	MQINQ
zdarzenie lokalne			ZDARZENIE MQIA_LOCAL_EVENT	MQINQ
zdarzenie programu rejestrującego			ZDARZENIEM MQIA_LOGGER_EVENT	MQINQ
Nazwa grupy LU			MQCA_LU_GROUP_NAME	MQINQ
Nazwa LU			MQCA_LU_NAME	MQINQ
Przyrostek ramienia lu62			MQCA_LU62_ARM_SUFFIX	MQINQ
Kanały lu62			MQIA_LU62_CHANNELS	MQINQ
maksymalna liczba aktywnych kanałów			MQIA_ACTIVE_CHANNELS,	MQINQ
Maksimum kanałów			MQIA_MAX_CHANNELS	MQINQ
maksymalna liczba uchwytów			MQIA_MAX_UCHWYTY	MQINQ
Maksymalna długość komunikatu			MQIA_MAX_MSG_LENGTH	MQINQ
maksymalny priorytet			MQIA_MAX_PRIORITY	MQINQ

Tabela 863. Atrybuty, struktury danych, pola, zapytania i wywołania (kontynuacja)

Atrybut	Struktura danych	Pole	Zapytanie	Zadzwoń
Maks. liczba niezatw. kom.			MQIA_MAX_UNCOMMITTED_MSGS	MQINQ
Rozliczanie MQI			LICZNIK MQIA_ACCOUNTING_MQI	MQINQ
Statystyka MQI			MQIA_STATISTICS_MQI	MQINQ
maksymalna liczba portów wychodzących			MQIA_OUTBOUND_PORT_MAX	MQINQ
Minimalny port danych wychodzących			MQIA_OUTBOUND_PORT_MIN	MQINQ
Hasło	Protokół MQCSP	CSPPasswordPtr		MQCONNX
	Protokół MQCSP	CSPPasswordOffset		MQCONNX
	Protokół MQCSP	CSPPasswordLength		MQCONNX
zdarzenie dotyczące wydajności			MQIA_PERFORMANCE_EVENT	MQINQ
rzeczy			MQIA_PLATFORMA	MQINQ
Rozliczanie kolejek			LICZNIK MQIA_ACTING_Q	MQINQ
Monitorowanie kolejek			MQIA_MONITORING_Q (kolejka MQIA)	MQINQ
Statystyka kolejek			MQIA_STATISTICS_Q (kolejka MQIA)	MQINQ
Limit czasu odbierania			LIMIT_CZASU_ODBIORU_MQIA	MQINQ
Minimalny limit czasu odbioru			MQIA_RECEIVE_TIMEOUT_MIN	MQINQ
Typ limitu czasu odbierania			MQIA_RECEIVE_TIMEOUT_TYPE	MQINQ
zdarzenie zdalne			MQIA_REMOTE_EVENT (zdarzenie MQIA)	MQINQ
Nazwa repozytorium			MQCA_NAZWA_REPOZYTORIUM	MQINQ
Lista nazw repozytorium			MQCA_REPOSITORY_NAMELIST,	MQINQ
nazwa menedżera kolejek współużytkowanych			MQIA_SHARED_Q_Q_MGR_NAME	MQINQ
Zdarzenie SSL			MQIA_SSL_EVENT	MQINQ

Tabela 863. Atrybuty, struktury danych, pola, zapytania i wywołania (kontynuacja)				
Atrybut	**Struktura danych**	**Pole**	**Zapytanie**	**Zadzwoń**
Ssl fips			MQIA_SSL_FIPS_REQUIRED	MQINQ
Licznik zerowania klucza SSL			MQIA_SSL_RESET_COUNT,	MQINQ
Zdarzenie uruchomienia i zatrzymania			MQIA_START_STOP_EVENT	MQINQ
Interwał statystyki			MQIA_STATISTICS_INTERVAL	MQINQ
Dostępność punktu synchronizacji			Punkt SYNCPOINT MQIA_SYNCPOINT	MQINQ
kanały tcp			MQIA_TCP_CHANNELS,	MQINQ
Podtrzymuj połączenie TCP			MQIA_TCP_KEEP_ALIVE	MQINQ
Nazwa TCP			MQCA_NAZWA_TCP	MQINQ
Typ stosu TCP			MQIA_TCP_STACK_TYPE	MQINQ
Zapis śledzenia trasy			MQIA_TRACE_ROUTE_RECORDING (zapis trasy MQ)	MQINQ
Interwał wyzwalacza			MQIA_TRIGGER_INTERVAL	MQINQ
ID użytkownika	Protokół MQCSP	CSPUserIdPtr		MQCONNX
	Protokół MQCSP	CSPUserIdprzesunięcie		MQCONNX
	Protokół MQCSP	CSPUserIdDługość		MQCONNX

ImqReferenceOdwołanie do nagłówka

Odwołanie do atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord w języku C + +.

Tabela 864. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
środowisko docelowe	MQRMH	DestEnvDługość, DestEnvPrzesunięcie
Nazwa miejsca docelowego	MQRMH	DestNameDługość, DestNamePrzesunięcie
Identyfikator instancji	MQRMH	Identyfikator ObjectInstance
długość logiczna	MQRMH	Długość DataLogical
przesunięcie logiczne	MQRMH	Przesunięcie DataLogical
przesunięcie logiczne 2	MQRMH	DataLogicalOffset2
Typ odwołania	MQRMH	ObjectType
Środowisko źródłowe	MQRMH	SrcEnvDługość, Przesunięcie SrcEnv

Tabela 864. Odwzorowanie atrybutów, struktur danych i pól (kontynuacja)		
Atrybut	**Struktura danych**	**Pole**
NAZWA ŹRÓDŁA	MQRMH	SrcNameDługość, SrcNamePrzesunięcie

Odwołanie ImqTrigger

Odwołanie do atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord w języku C + +.

Tabela 865. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
Identyfikator aplikacji	MQTM	ApplId
Typ aplikacji	MQTM	ApplType
Dane środowiska	MQTM	EnvData
Nazwa procesu	MQTM	ProcessName
Nazwa kolejki	MQTM	Nazwa QName
Dane wyzwalacza	MQTM	TriggerData
Dane użytkownika	MQTM	UserData

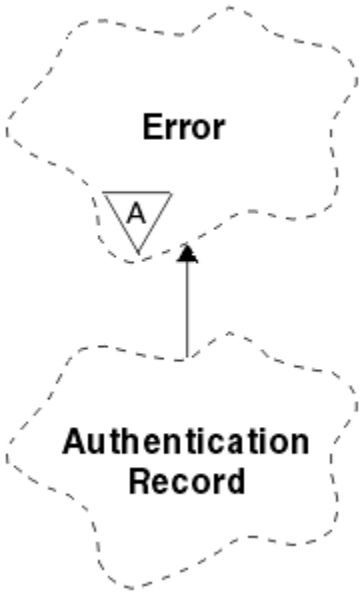
Odwołanie do nagłówka ImqWork

Odwołanie do atrybutów, struktur danych i pól dla klasy ImqAuthenticationRecord w języku C + +.

Tabela 866. Odwzorowanie atrybutów, struktur danych i pól		
Atrybut	**Struktura danych**	**Pole**
znacznik komunikatu	MQWIH	MessageToken
Nazwa usługi	MQWIH	ServiceName
krok usługi	MQWIH	ServiceStep

Klasa ImqAuthenticationRecord języka C++

Ta klasa hermetyzuje rekord informacji uwierzytelniającej (MQAIR) do użycia podczas wykonywania menedżera ImqQueue: metoda connect dla niestandardowych połączeń klienta TLS.



Rysunek 14. Klasa rekordu ImqAuthentication

Więcej szczegółów zawiera opis metody ImqQueueManager: :connect. Ta klasa nie jest dostępna na platformie z/OS .

- "Atrybuty obiektu" na stronie 1858
- "Konstruktory" na stronie 1859
- "Metody obiektów (publiczne)" na stronie 1859
- "Metody obiektów (zabezpieczone)" na stronie 1859

Atrybuty obiektu

Typ połączenia
Nazwa połączenia z serwerem CRL LDAP. Jest to adres IP lub nazwa DNS, po której opcjonalnie znajduje się numer portu w nawiasach.

odwołanie do połączenia
Odwołanie do obiektu menedżera ImqQueue, który udostępnia wymagane połączenie z (lokalnym) menedżerem kolejek. Wartością początkową jest zero. Nie należy mylić tego elementu z nazwą menedżera kolejek, która identyfikuje menedżer kolejek (prawdopodobnie zdalny) dla nazwanej kolejki.

następny rekord uwierzytelniania
Następny obiekt tej klasy, bez określonej kolejności, mający takie samo **odwołanie do połączenia** jak ten obiekt. Wartością początkową jest zero.

Hasło
Hasło podane na potrzeby uwierzytelniania połączenia z serwerem CRL LDAP.

poprzedni rekord uwierzytelniania
Poprzedni obiekt tej klasy, bez określonej kolejności, mający takie samo **odwołanie do połączenia** jak ten obiekt. Wartością początkową jest zero.

typ
Typ informacji uwierzytelniających zawartych w rekordzie.

nazwa użytkownika,
Identyfikator użytkownika podany w celu autoryzacji na serwerze CRL LDAP.

Konstruktory

ImqAuthenticationRecord ();
Konstruktor domyślny.

Metody obiektów (publiczne)

void operator = (const ImqAuthenticationRecord & ***air*** **);**
Kopiuje dane instancji z *air*, zastępując istniejące dane instancji.

const ImqString & connectionName () konst;
Zwraca **nazwę połączenia**.

void setConnectionName (const ImqString & ***nazwa*** **);**
Ustawia **nazwę połączenia**.

void setConnectionNazwa (const char * ***nazwa*** **= 0);**
Ustawia **nazwę połączenia**.

ImqQueueManager * connectionReference () konst;
Zwraca **odwołanie do połączenia**.

void setConnectionReference (ImqQueueManager & ***menedżer*** **);**
Ustawia **odwołanie do połączenia**.

void setConnectionReference (ImqQueueManager * ***menedżer*** **= 0);**
Ustawia **odwołanie do połączenia**.

void copyOut (MQAIR * ***pAir*** **);**
Kopiuje dane instancji do *pAir*, zastępując istniejące dane instancji. Może to wymagać przydzielenia zależnej pamięci masowej.

void clear (MQAIR * ***pAir*** **);**
Czyści strukturę i zwalnia zależną pamięć masową, do której odwołuje się *pAir*.

Rekord ImqAuthentication* nextAuthenticationRecord () konst;
Zwraca **następny rekord uwierzytelniania**.

const ImqString & password () konst;
Zwraca **hasło**.

void setPassword (const ImqString & ***hasło*** **);**
Ustawia **hasło**.

void setPassword (const char * ***hasło*** **= 0);**
Ustawia **hasło**.

Rekord ImqAuthentication* previousAuthenticationRecord () konst;
Zwraca **poprzedni rekord uwierzytelniania**.

MQLONG type () (typ MQLONG) konst;
Zwraca **typ**.

void setType (const MQLONG ***typ*** **);**
Ustawia **typ**.

const ImqString & userName () konst;
Zwraca **nazwę użytkownika**.

void setUserName (const ImqString & ***nazwa*** **);**
Ustawia **nazwę użytkownika**.

void setUserNazwa (const char * ***nazwa*** **= 0);**
Ustawia **nazwę użytkownika**.

Metody obiektów (zabezpieczone)

void setNextAuthenticationRecord (ImqAuthenticationRecord * ***pAir*** **= 0);**
Ustawia **następny rekord uwierzytelniania**.

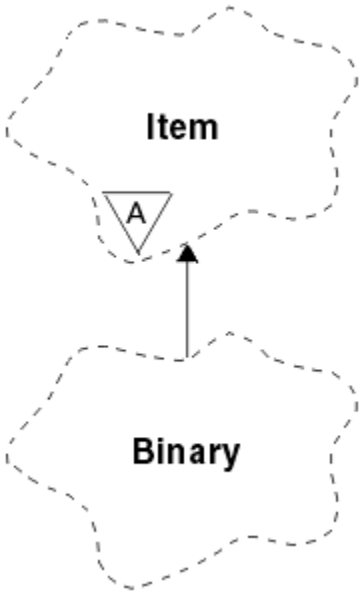
Uwaga: Funkcji tej należy używać tylko wtedy, gdy istnieje pewność, że nie złamie ona listy rekordów uwierzytelniania.

void setPreviousAuthenticationRecord (ImqAuthenticationRecord * *pAir* = 0);
Ustawia **poprzedni rekord uwierzytelniania**.

Uwaga: Funkcji tej należy używać tylko wtedy, gdy istnieje pewność, że nie złamie ona listy rekordów uwierzytelniania.

Klasa C++ ImqBinary

Ta klasa hermetyzuje binarną tablicę bajtów, która może być używana dla wartości ImqMessage **token rozliczania**, **identyfikator korelacji**i **identyfikator komunikatu** . Umożliwia łatwe przypisywanie, kopiowanie i porównywanie.



Rysunek 15. Klasa ImqBinary

- "Atrybuty obiektu" na stronie 1860
- "Konstruktory" na stronie 1860
- "Przeciążone metody ImqItem" na stronie 1861
- "Metody obiektów (publiczne)" na stronie 1861
- "Metody obiektów (zabezpieczone)" na stronie 1861
- "Kody przyczyny" na stronie 1861

Atrybuty obiektu

jakościowe
Tablica bajtów danych binarnych. Wartość początkowa jest pusta.

Długość danych
Liczba bajtów. Wartością początkową jest zero.

wskaźnik danych
Adres pierwszego bajtu **danych**. Wartością początkową jest zero.

Konstruktory

ImqBinary( );
Konstruktor domyślny.

ImqBinary(const ImqBinary & ***binary*** **);**
Konstruktor kopii.

ImqBinary(const void * ***dane*****, const size_t** ***długość*** **);**
Kopiuje *długość* bajtów z *danych*.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & ***komunikat*** **);**
Kopiuje **dane** do buforu komunikatów, zastępując istniejącą treść. Ustawia parametr *msg* **format** na wartość MQFMT_NONE.

Więcej informacji na ten temat zawiera opis metody klasy ImqItem .

wirtualny ImqBoolean pasteIn (ImqMessage & ***komunikat*** **);**
Ustawia **dane** , przesyłając pozostałe dane z buforu komunikatów, zastępując istniejące **dane**.

Aby operacja zakończyła się pomyślnie, **format** komunikatu ImqMessage musi mieć wartość MQFMT_NONE.

Więcej informacji na ten temat zawiera opis metody klasy ImqItem .

Metody obiektów (publiczne)

void operator = (const ImqBinary & ***binary*** **);**
Kopiuje bajty z pliku *binarnego*.

ImqBoolean operator == (const ImqBinary & ***binary*** **);**
Porównuje ten obiekt z obiektem *binary*. W przeciwnym razie zwraca FALSE jeśli nie jest równe lub TRUE. Obiekty są równe, jeśli mają taką samą **długość danych** i są zgodne z bajtami.

ImqBoolean copyOut (void * ***buffer*****, const size_t** ***length*****, const char** ***pad*** **= 0);**
Kopiuje do *długości* bajtów z **wskaźnika danych** do *buforu*. Jeśli **długość danych** jest niewystarczająca, pozostała przestrzeń w *buforze* jest wypełniona *dopełnieniem* bajtów. Parametr *buffer* może mieć wartość zero, jeśli parametr *length* ma również wartość zero. *długość* nie może być ujemna. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

size_t dataLength () const (konst) ;
Zwraca **długość danych**.

ImqBoolean setDataDługość (const size_t ***długość*** **);**
Ustawia **długość danych**. Jeśli w wyniku tej metody zmieniona zostanie **długość danych** , dane w obiekcie nie zostaną zainicjowane. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

void * dataPointer () const (konst) ;
Zwraca **wskaźnik danych**.

ImqBoolean isNull () const (konst) ;
Zwraca wartość PRAWDA, jeśli **długość danych** wynosi zero lub jeśli wszystkie bajty **danych** są równe zero. W przeciwnym razie zwraca FALSE.

ImqBoolean set (const void * ***buffer*****, const size_t** ***długość*** **);**
Kopiuje *długość* bajtów z *buforu*. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Metody obiektów (zabezpieczone)

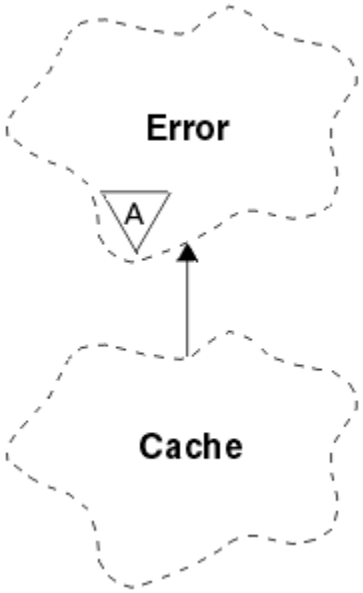
void clear ();
Zmniejsza **długość danych** do zera.

Kody przyczyny

- MQRC_NO_BUFFER
- MQRC_STORAGE_NIEDOSTĘPNY
- MQRC_INCONSISTENT_FORMAT,

Klasa ImqCache C++

Ta klasa służy do przechowywania lub zestawiania danych w pamięci.



Rysunek 16. Klasa ImqCache

Ta klasa służy do przechowywania lub zestawiania danych w pamięci. Można wyznaczyć bufor pamięci o stałej wielkości lub system może automatycznie udostępnić elastyczną ilość pamięci. Ta klasa odnosi się do wywołań MQI wymienionych w sekcji "Odwołanie ImqCache" na stronie 1842.

Atrybuty obiektu

bufor automatyczny
Wskazuje, czy pamięć buforu jest zarządzana automatycznie przez system (TRUE), czy też jest dostarczana przez użytkownika (FALSE). Początkowo ma wartość TRUE.

Ten atrybut nie jest ustawiony bezpośrednio. Jest ona ustawiana pośrednio przy użyciu metody **useEmptyBuffer** lub **useFullBuffer** .

Jeśli podano pamięć masową użytkownika, atrybut ten ma wartość FALSE, pamięć buforu nie może rosnąć i mogą wystąpić błędy przepełnienia buforu. Adres i długość buforu pozostają stałe.

Jeśli pamięć masowa użytkownika nie zostanie podana, atrybut ten będzie mieć wartość TRUE, a pamięć buforu może rosnąć przyrostowo, aby pomieścić dowolną ilość danych komunikatu. Jednak gdy bufor rośnie, adres buforu może się zmienić, dlatego należy zachować ostrożność podczas używania **wskaźnika buforu** i **wskaźnika danych**.

długość buforu
Liczba bajtów pamięci w buforze. Wartością początkową jest zero.

wskaźnik buforu
Adres pamięci buforu. Wartość początkowa jest pusta.

Długość danych
Liczba bajtów następujących po **wskaźniku danych**. Wartość ta musi być równa lub mniejsza niż **długość komunikatu**. Wartością początkową jest zero.

Pozycja danych
Liczba bajtów poprzedzających **wskaźnik danych**. Wartość ta musi być równa lub mniejsza niż **długość komunikatu**. Wartością początkową jest zero.

wskaźnik danych
Adres części buforu, która ma zostać zapisana lub odczytana z następnego buforu. Wartość początkowa jest pusta.

długość komunikatu
Liczba bajtów istotnych danych w buforze. Wartością początkową jest zero.

Konstruktory

ImqCache( );
Konstruktor domyślny.

ImqCache(const ImqCache & ***cache*** **);**
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqCache & ***cache*** **);**
Kopiuje do obiektu maksymalnie **długość komunikatu** bajtów danych z obiektu *cache* . Jeśli **bufor automatyczny** ma wartość FALSE, **długość buforu** musi być wystarczająca, aby pomieścić skopiowane dane.

ImqBoolean automaticBuffer () const (konst) ;
Zwraca wartość **automatycznego buforu** .

size_t bufferLength () const (konst) ;
Zwraca **długość buforu**.

char * bufferPointer () const (konst) ;
Zwraca **wskaźnik buforu**.

void clearMessage ();
Ustawia **długość komunikatu** i **przesunięcie danych** na zero.

size_t dataLength () const (konst) ;
Zwraca **długość danych**.

size_t dataOffset () const (konst) ;
Zwraca **przesunięcie danych**.

ImqBoolean setDataprzesunięcie (const size_t ***przesunięcie*** **);**
Ustawia **przesunięcie danych**. W razie potrzeby zwiększana jest **długość komunikatu** , aby zapewnić, że nie będzie ona mniejsza niż **przesunięcie danych**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

char * dataPointer () const (konst) ;
Zwraca kopię **wskaźnika danych**.

size_t messageLength () const (konst) ;
Zwraca **długość komunikatu**.

ImqBoolean setMessageDługość (const size_t ***długość*** **);**
Ustawia **długość komunikatu**. W razie potrzeby zwiększa **długość buforu** , aby upewnić się, że **długość komunikatu** nie jest większa niż **długość buforu**. W razie potrzeby zmniejsza **przesunięcie danych** , aby zapewnić, że nie jest ono większe niż **długość komunikatu**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean moreBytes (const size_t ***bytes-required*** **);**
Zapewnia, że między **wskaźnikiem danych** a końcem buforu będzie dostępny (do zapisu) więcej bajtów *wymaganych bajtów* . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Jeśli **bufor automatyczny** ma wartość TRUE, uzyskiwana jest większa ilość pamięci zgodnie z wymaganiami; w przeciwnym razie **długość buforu** musi być już odpowiednia.

ImqBoolean odczyt (const size_t *długość*, char * & *external-buffer*);
Kopiuje *długość* bajtów z buforu począwszy od pozycji **wskaźnika danych** do *buforu zewnętrznego*. Po skopiowaniu danych **przesunięcie danych** jest zwiększane o *długość*. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean resizeBuffer (const size_t *długość*);
Zmienia **długość buforu**, pod warunkiem, że **automatyczny bufor** ma wartość TRUE. W tym celu należy zmienić przydział pamięci buforu. Do **długości komunikatu** bajtów danych z istniejącego buforu są kopiowane do nowego buforu. Maksymalna liczba kopiowanych bajtów wynosi *długość* . **Wskaźnik buforu** został zmieniony. **Długość komunikatu** i **przesunięcie danych** są zachowywane w miarę możliwości w granicach nowego buforu. Zwraca TRUE, jeśli operacja się powiodła, lub FALSE, jeśli **automatyczny bufor** ma wartość FALSE.

Uwaga: Ta metoda może zakończyć się niepowodzeniem z błędem MQRC_STORAGE_NOT_AVAILABLE, jeśli wystąpi problem z zasobami systemowymi.

ImqBoolean useEmptyBuffer (const char * *external-buffer*, const size_t *długość*);
Identyfikuje pusty bufor użytkownika, ustawiając **wskaźnik buforu** tak, aby wskazywał na *bufor zewnętrzny*, **długość buforu** na *długości* **długość komunikatu** na zero. Wykonuje operację **clearMessage**. Jeśli bufor jest w pełni zapełniony danymi, należy użyć metody **useFullBuffer** . Jeśli bufor jest częściowo zapełniony danymi, należy użyć metody **setMessageLength** w celu wskazania poprawnej wartości. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Ta metoda może być używana do identyfikowania stałej ilości pamięci, zgodnie z wcześniejszym opisem (*bufor zewnętrzny* nie jest pusty, a *długość* jest niezerowa), w którym to przypadku **bufor automatyczny** jest ustawiony na wartość FALSE lub może być używany do przywracania pamięci elastycznej zarządzanej przez system (*bufor zewnętrzny* jest pusty, a *długość* wynosi zero), w którym to przypadku **bufor automatyczny** jest ustawiony na wartość TRUE.

ImqBoolean useFullBuffer (const char * *externalBuffer*, const size_t *długość*);
Podobnie jak w przypadku buforu **useEmptyBuffer**, z tą różnicą, że **długość komunikatu** jest ustawiona na wartość *length*. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean zapis (const size_t *długość*, const char * *external-buffer*);
Kopiuje *długość* bajtów z *buforu zewnętrznego*do buforu począwszy od pozycji **wskaźnika danych** . Po skopiowaniu danych **przesunięcie danych** jest zwiększane o *długość*, a **długość komunikatu** jest zwiększana, jeśli jest to konieczne, w celu zapewnienia, że nie jest mniejsza niż nowa wartość **przesunięcia danych** . Ta metoda zwraca wartość TRUE w przypadku powodzenia.

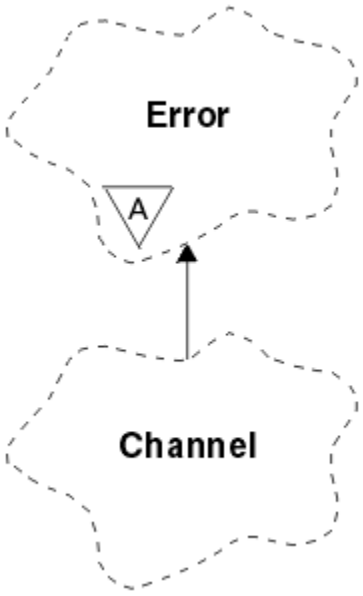
Jeśli **bufor automatyczny** ma wartość TRUE, gwarantowana jest odpowiednia ilość pamięci; w przeciwnym razie największe **przesunięcie danych** nie może przekraczać **długości buforu**.

Kody przyczyny

- MQRC_BUFFER_NOT_AUTOMATIC
- MQRC_DATA_OBCIĘTE
- MQRC_INSUFFICIENT_BUFFER (MQRC_INC_BUFFER)
- MQRC_INSUFFICIENT_DATA
- MQRC_NULL_POINTER (wskaźnik wartości NULL)
- MQRC_STORAGE_NIEDOSTĘPNY
- MQRC_ZERO_LENGTH

Klasa ImqChannel C++

Ta klasa hermetyzuje definicję kanału (MQCD) do użycia podczas wykonywania metody Manager: :connect dla niestandardowych połączeń klienckich.



Rysunek 17. Klasa ImqChannel

Więcej szczegółowych informacji można znaleźć w opisie metody Manager: :connect i przykładowego programu HELLO WORLD (imqwrld.cpp).

Nie wszystkie wymienione metody mają zastosowanie do wszystkich platform. Więcej informacji zawierają opisy komend DEFINE CHANNEL i ALTER CHANNEL .

Klasa ImqChannel nie jest obsługiwana w systemie z/OS.

Atrybuty obiektu

bicie serca
Liczba milisekund między operacjami sprawdzenia, czy kanał zdalny jest aktywny. Wartością początkową jest 0.

nazwa kanału
Nazwa kanału. Wartość początkowa jest pusta.

Typ połączenia
Nazwa połączenia. Na przykład adres IP hosta. Wartość początkowa jest pusta.

Kompresja nagłówka
Lista technik kompresji danych nagłówka obsługiwanych przez kanał. Wszystkie wartości początkowe są ustawione na wartość MQCOMPRESS_NOT_AVAILABLE.

interwał pulsu
Liczba sekund między sprawdzeniami, czy połączenie nadal działa. Wartością początkową jest 300.

Interwał sprawdzania połączenia
Liczba sekund, które zostały przekazane do stosu komunikacyjnego, określająca czas utrzymywania aktywności dla kanału. Wartością początkową jest MQKAI_AUTO.

Adres lokalny
Adres komunikacji lokalnej dla kanału.

Maksymalna długość komunikatu
Maksymalna długość komunikatu obsługiwana przez kanał w pojedynczej komunikacji. Wartością początkową jest 4 194 304.

Kompresja komunikatu
Lista technik kompresji danych komunikatu obsługiwanych przez kanał. Wszystkie wartości początkowe są ustawione na wartość MQCOMPRESS_NOT_AVAILABLE.

Nazwa trybu
Nazwa trybu. Wartość początkowa jest pusta.

Hasło
Hasło podane na potrzeby uwierzytelniania połączenia. Wartość początkowa jest pusta.

liczba wyjść odbierania
Liczba wyjść odbierania. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

nazwy wyjść odbierania
Nazwy wyjść odbierania.

odbierz dane użytkownika
Dane powiązane z wyjściami odbierania.

Nazwa wyjścia zabezpieczeń
Nazwa wyjścia zabezpieczeń, które ma być wywołane po stronie serwera połączenia. Wartość początkowa jest pusta.

dane użytkownika dotyczące bezpieczeństwa
Dane, które mają zostać przekazane do wyjścia zabezpieczeń. Wartość początkowa jest pusta.

liczba wyjść wysyłania
Liczba wyjść wysyłania. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

nazwy wyjść wysyłania
Nazwy wyjść wysyłania.

wyślij dane użytkownika
Dane powiązane z wyjściami wysyłania.

CipherSpec SSL
CipherSpec do użycia z protokołem TLS.

Typ uwierzytelniania klienta SSL
Typ uwierzytelniania klienta do użycia z protokołem TLS.

Nazwa węzła sieci SSL
Nazwa węzła sieci do użycia z protokołem TLS.

Nazwa programu transakcyjnego
Nazwa programu transakcyjnego. Wartość początkowa jest pusta.

Typ transportu
Typ transportu połączenia. Wartością początkową jest MQXPT_LU62.

ID użytkownika
Identyfikator użytkownika podany dla autoryzacji. Wartość początkowa jest pusta.

Konstruktory

ImqChannel( ) ;
Konstruktor domyślny.

ImqChannel(const ImqChannel & *kanał*);
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqChannel & *kanał*);
Kopiuje dane instancji z *kanału*, zastępując istniejące dane instancji.

MQLONG batchHeartBeat () konst;
Zwraca **puls zadania wsadowego**.

ImqBoolean setBatchHeartBeat(const MQLONG *puls* = 0L);
Ustawia **puls zadania wsadowego**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString channelName() konst;
Zwraca **nazwę kanału**.

ImqBoolean setChannelNazwa (const char * *nazwa* = 0);
Ustawia **nazwę kanału**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString connectionName() konst;
Zwraca **nazwę połączenia**.

ImqBoolean setConnectionNazwa (const char * *nazwa* = 0);
Ustawia **nazwę połączenia**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

size_t headerCompressionLiczba () konst;
Zwraca liczbę obsługiwanych technik kompresji danych nagłówka.

ImqBoolean headerCompression(liczba wielkości konst, MQLONG compress []) konst;
Zwraca kopie obsługiwanych technik kompresji danych nagłówka w **compress**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setHeaderCompression (const size_t count, const MQLONG compress []);

Ustawia obsługiwane techniki kompresji danych nagłówka na **compress**.

Ustawia obsługiwaną metodę kompresji danych nagłówka na **count**.

Ta metoda zwraca wartość TRUE w przypadku powodzenia.

MQLONG heartBeatInterwał () konst;
Zwraca **interwał pulsu**.

ImqBoolean setHeartBeatInterval(const MQLONG *interwał* = 300L);
Ustawia **interwał pulsu**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Odstęp czasu MQLONG keepAlive() konst;
Zwraca **interwał sprawdzania połączenia**.

ImqBoolean setKeepAliveInterval(const MQLONG *interwał* = MQKAI_AUTO);
Ustawia **interwał sprawdzania połączenia**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString localAddress() const;
Zwraca **adres lokalny**.

ImqBoolean setLocalAdres (const char * *adres* = 0);
Ustawia **adres lokalny**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

MQLONG maximumMessageLength () konst;
Zwraca **maksymalną długość komunikatu**.

ImqBoolean setMaximumMessageLength(const MQLONG *długość* = 4194304L);
Ustawia **maksymalną długość komunikatu**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

size_t messageCompressionLiczba () konst;
Zwraca liczbę obsługiwanych technik kompresji danych komunikatu.

ImqBoolean messageCompression(const size_t count, MQLONG compress []) const;
Zwraca kopie obsługiwanych technik kompresji danych komunikatu w **compress**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setMessageKompresja (const size_t licznik, const MQLONG compress []);

Ustawia obsługiwane techniki kompresji danych komunikatu, które mają być kompresowane.

Ustawia liczbę obsługiwanych technik kompresji danych komunikatu na wartość count.

Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString modeName() konst;
Zwraca **nazwę trybu**.

ImqBoolean setModeNazwa (const char * *nazwa* = 0);
Ustawia **nazwę trybu**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString password () konst;
Zwraca **hasło**.

ImqBoolean setPassword(const char * *hasło* = 0);
Ustawia **hasło**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

size_t receiveExitLiczba () konst;
Zwraca **licznik wyjść odbierania**.

ImqString receiveExitNazwa ();
Zwraca pierwszą z **nazw wyjść odbierania**, jeśli istnieją. Jeśli **liczba wyjść odbierania** wynosi zero, zwraca pusty łańcuch.

ImqBoolean receiveExitNames (const size_t *count*, ImqString * *names* []);
Zwraca kopie **nazw wyjść odbierania** w *nazwach*. Ustawia wszystkie *nazwy* przekraczające **liczbę wyjść odbierania** na łańcuchy o wartości NULL. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setReceiveExitName(const char * *nazwa* = 0);
Ustawia **nazwy wyjść odbierania** na pojedynczą *nazwę*. *nazwa* może być pusta lub mieć wartość null. Ustawia **liczbę wyjść odbierania** na 1 lub zero. Usuwa zawartość pola **Odbierz dane użytkownika**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setReceiveExitNames(const size_t *count*, const char * *names* []);
Ustawia **nazwy wyjść odbierania** na *names*. Poszczególne wartości *names* nie mogą być puste ani mieć wartości NULL. Ustawia **liczbę wyjść odbierania** na *count*. Usuwa zawartość pola **Odbierz dane użytkownika**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setReceiveExitNames(const size_t *count*, const ImqString * *names* []);
Ustawia **nazwy wyjść odbierania** na *names*. Poszczególne wartości *names* nie mogą być puste ani mieć wartości NULL. Ustawia **liczbę wyjść odbierania** na *count*. Usuwa zawartość pola **Odbierz dane użytkownika**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString receiveUserDane ();
Zwraca pierwszy z elementów **odbierania danych użytkownika** (jeśli istnieją). Jeśli **liczba wyjść odbierania** wynosi zero, zwraca pusty łańcuch.

ImqBoolean receiveUserDane (const size_t *liczba*, ImqString * *dane* []);
Zwraca kopie elementów **odbierania danych użytkownika** w *danych*. Ustawia *dane* przekraczające **liczbę wyjść odbierania** na łańcuchy o wartości NULL. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setReceiveUserData(const char * *dane* = 0);
Ustawia **odbieranie danych użytkownika** na pojedynczy element *data*. Jeśli parametr *data* nie ma wartości null, **liczba wyjść odbierania** musi wynosić co najmniej 1. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setReceiveUserData(const size_t *liczba*, const char * *dane* []);
Ustawia **odbieranie danych użytkownika** na *dane*. *liczba* nie może być większa niż **liczba wyjść odbierania**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setReceiveUserData(const size_t *liczba*, const ImqString * *dane* []);
Ustawia **odbieranie danych użytkownika** na *dane*. *liczba* nie może być większa niż **liczba wyjść odbierania**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString securityExitName () konst;
Zwraca **nazwę wyjścia zabezpieczeń**.

ImqBoolean setSecurityExitName(const char * *nazwa* = 0);
Ustawia **nazwę wyjścia zabezpieczeń**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString securityUserData () konst;
Zwraca **dane użytkownika zabezpieczeń**.

ImqBoolean setSecurityUserData(const char * *dane* = 0);
Ustawia **dane użytkownika dotyczące bezpieczeństwa**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

size_t sendExitLiczba () konst;
Zwraca **licznik wyjść wysyłania**.

ImqString sendExitNazwa ();
Zwraca pierwszą z **nazw wyjść wysyłania**, jeśli istnieją. Zwraca pusty łańcuch, jeśli **liczba wyjść wysyłania** wynosi zero.

ImqBoolean sendExitNames (const size_t *count*, ImqString * *names* []);
Zwraca kopie **nazw wyjść wysyłania** w polu *nazwy*. Ustawia *nazwy* przekraczające **liczbę wyjść wysyłania** na łańcuchy o wartości NULL. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setSendExitName(const char * *nazwa* = 0);
Ustawia **nazwy wyjść wysyłania** na jedną *nazwę*. *nazwa* może być pusta lub mieć wartość null. Ustawia **liczbę wyjść wysyłania** na 1 lub zero. Usuwa zawartość pola **send user data**(Wyślij dane użytkownika). Ta metoda zwraca wartość PRAWDA, jeśli powiedzie się

ImqBoolean setSendExitNames(const size_t *count*, const char * *names* []);
Ustawia **nazwy wyjść wysyłania** na *names*. Poszczególne wartości *names* nie mogą być puste ani mieć wartości NULL. Ustawia **liczbę wyjść wysyłania** na *count*. Usuwa zawartość pola **send user data**(Wyślij dane użytkownika). Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setSendExitNames(const size_t *count*, const ImqString * *names* []);
Ustawia **nazwy wyjść wysyłania** na *names*. Poszczególne wartości *names* nie mogą być puste ani mieć wartości NULL. Ustawia **liczbę wyjść wysyłania** na *count*. Usuwa zawartość pola **send user data**(Wyślij dane użytkownika). Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString sendUserDane ();
Zwraca pierwszy z elementów **wysyłania danych użytkownika** , jeśli istnieją. Zwraca pusty łańcuch, jeśli **liczba wyjść wysyłania** wynosi zero.

ImqBoolean sendUserData (const size_t *liczba*, ImqString * *dane* []);
Zwraca kopie elementów **wysyłania danych użytkownika** w *danych*. Ustawia *dane* przekraczające **liczbę wyjść wysyłania** na łańcuchy o wartości NULL. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setSendUserData(const char * *dane* = 0);
Ustawia **wysyłanie danych użytkownika** do pojedynczego elementu *data*. Jeśli parametr *data* nie ma wartości null, **liczba wyjść wysyłania** musi wynosić co najmniej 1. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setSendUserData(const size_t *count*, const char * *data* []);
Ustawia **wysyłanie danych użytkownika** na *dane*. *liczba* nie może być większa niż **liczba wyjść wysyłania**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setSendUserData(const size_t *liczba*, const ImqString * *dane* []);
Ustawia **wysyłanie danych użytkownika** na *dane*. *liczba* nie może być większa niż **liczba wyjść wysyłania**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString sslCipherSpecification () konst;
Zwraca specyfikację szyfru TLS.

ImqBoolean setSslCipherSpecification(const char * *nazwa* = 0);
Ustawia specyfikację szyfru TLS. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

MQLONG sslClient, Uwierzytelnianie () konst;
Zwraca typ uwierzytelniania klienta TLS.

ImqBoolean setSslClientAuthentication(const MQLONG *auth* = MQSCA_REQUIRED);
Ustawia typ uwierzytelniania klienta TLS. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString sslPeerName () konst;
Zwraca nazwę węzła TLS.

ImqBoolean setSslPeerName(const char * *nazwa* = 0);
Ustawia nazwę węzła TLS. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString transactionProgramNazwa () konst;
Zwraca **nazwę programu transakcyjnego**.

ImqBoolean setTransactionProgramName(const char * *nazwa* = 0);
Ustawia **nazwę programu transakcyjnego**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

MQLONG transportType() konst;
Zwraca **typ transportu**.

ImqBoolean setTransportTyp (const MQLONG *typ* = MQXPT_LU62);
Ustawia **typ transportu**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString userId() konst;
Zwraca **identyfikator użytkownika**.

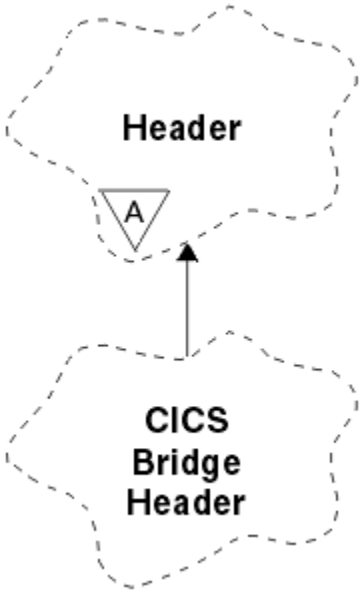
ImqBoolean setUserID (const char * *id* = 0);
Ustawia **ID użytkownika**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Kody przyczyny

- BŁĄD MQRC_DATA_LENGTH_ERROR
- BŁĄD MQRC_ITEM_COUNT_ERROR
- MQRC_NULL_POINTER (wskaźnik wartości NULL)
- BŁĄD MQRC_SOURCE_BUFFER_ERROR

ImqCICSBridgeHeader (klasa C + +)

Ta klasa hermetyzuje konkretne funkcje struktury danych MQCIH.



Rysunek 18. Imq, klasaCICSBridgeHeader

Obiekty tej klasy są używane przez aplikacje, które wysyłają komunikaty do CICS bridge za pośrednictwem IBM MQ for z/OS.

- "Atrybuty obiektu" na stronie 1871
- "Konstruktory" na stronie 1873
- "Przeciążone metody ImqItem" na stronie 1873
- "Metody obiektów (publiczne)" na stronie 1873
- "Dane obiektu (zabezpieczone)" na stronie 1876

- “Kody przyczyny” na stronie 1876
- “Kody powrotu” na stronie 1876

Atrybuty obiektu

Deskryptor ADS
Deskryptor ADS wysyłania/odbierania. Jest on ustawiany za pomocą komendy MQCADSD_NONE. Wartością początkową jest MQCADSD_NONE. Możliwe są następujące wartości dodatkowe:

- MQCADSD_BRAK
- MQCADSD_SEND
- MQCADSD_RECV
- MQCADSD_MSGFORMAT

identyfikator uwagi
Klawisz AID. Pole musi mieć długość MQ_ATTENTION_ID_LENGTH.

element uwierzytelniający
Hasło lub przepustka RACF . Wartość początkowa zawiera odstępy o długości MQ_AUTHENTICATOR_LENGTH.

kod nieprawidłowego zakończenia mostu
Kod nieprawidłowego zakończenia mostu o długości MQ_ABEND_CODE_LENGTH. Wartością początkową są cztery znaki odstępu. Wartość zwracana w tym polu zależy od kodu powrotu. Więcej szczegółowych informacji zawiera sekcja Tabela 867 na stronie 1876.

kod anulowania mostu
Kod transakcji nieprawidłowego zakończenia mostu. Pole jest zastrzeżone, musi zawierać odstępy i mieć długość MQ_CANCEL_CODE_LENGTH.

kod zakończenia mostu
Kod zakończenia, który może zawierać kod zakończenia IBM MQ lub wartość EIBRESP CICS . Pole ma wartość początkową MQCC_OK. Wartość zwracana w tym polu zależy od kodu powrotu. Więcej szczegółowych informacji zawiera sekcja Tabela 867 na stronie 1876.

przesunięcie błędu mostu
Przesunięcie błędu mostu. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

kod przyczyny mostu
Kod przyczyny. To pole może zawierać przyczynę IBM MQ lub wartość CICS EIBRESP2 . Wartością początkową tego pola jest MQRC_NONE. Wartość zwracana w tym polu zależy od kodu powrotu. Więcej szczegółowych informacji zawiera sekcja Tabela 867 na stronie 1876.

kod powrotu mostu
Kod powrotu z CICS bridge. Wartością początkową jest MQCRC_OK.

zadanie konwersacyjne
Określa, czy zadanie może być konwersacyjne. Wartością początkową jest MQCCT_NO. Możliwe są następujące wartości dodatkowe:

- MQCCT_YES
- MQCCT_NO

pozycja kursora
Pozycja kursora. Wartością początkową jest zero.

czas przechowywania obiektu
Czas wydania narzędzia CICS bridge .

narzędzie podobne do
Emulowany atrybut terminalu. Pole musi mieć długość MQ_FACILITY_LIKE_LENGTH.

token narzędzia
Wartość tokenu BVT. Pole musi mieć długość MQ_FACILITY_LENGTH. Wartością początkową jest MQCFAC_NONE.

function (funkcja)
Funkcja, która może zawierać nazwę wywołania IBM MQ lub funkcję CICS EIBFN. Pole ma wartość początkową MQCFUNC_NONE o długości MQ_FUNCTION_LENGTH. Wartość zwracana w tym polu zależy od kodu powrotu. Więcej szczegółowych informacji zawiera sekcja Tabela 867 na stronie 1876.

Jeśli **funkcja** zawiera nazwę wywołania IBM MQ , możliwe są następujące wartości dodatkowe:

- MQCFUNC_MQCONN,
- MQCFUNC_MQGET,
- MQCFUNC_MQINQ,
- MQCFUNC_BRAK
- MQCFUNC_MQOPEN,
- MQCFUNC_PUT (MQCFUNC)
- MQCFUNC_MQPUT1

przedział czasu oczekiwania na pobranie
Odstęp czasu oczekiwania na wywołanie MQGET wydane przez zadanie CICS bridge . Wartością początkową jest MQCGWI_DEFAULT. To pole ma zastosowanie tylko wtedy, gdy **element sterujący jednostki pracy** ma wartość MQCUOWC_FIRST. Możliwe są następujące wartości dodatkowe:

- MQCGWI_DEFAULT
- MQWI_UNLIMITED

Typ odsyłacza
Typ powiązania. Wartością początkową jest MQCLT_PROGRAM. Możliwe są następujące wartości dodatkowe:

- MQCLT_PROGRAM
- MQCLT_TRANSACTION

identyfikator następnej transakcji
Identyfikator następnej transakcji do przyłączenia. Pole musi mieć długość MQ_TRANSACTION_ID_LENGTH.

długość danych wyjściowych
Długość danych COMMAREA. Wartością początkową jest MQCODL_AS_INPUT.

format odpowiedzi
Nazwa formatu komunikatu odpowiedzi. Wartością początkową jest MQFMT_NONE o długości MQ_FORMAT_LENGTH.

kod początkowy
Kod początkowy transakcji. Pole musi mieć długość MQ_START_CODE_LENGTH. Wartością początkową jest MQCSC_NONE. Możliwe są następujące wartości dodatkowe:

- MQCSC_START
- MQCSC_STARTDATA
- MQCSC_TERMINPUT
- MQCSC_BRAK

status zakończenia zadania
Status zakończenia zadania. Wartością początkową jest MQCTES_NOSYNC. Możliwe są następujące wartości dodatkowe:

- MQCTES_COMMIT
- MQCTES_BACKOUT
- MQCTES_ENDTASK
- MQCTES_NOSYNC

Identyfikator transakcji
Identyfikator transakcji do przyłączenia. Wartość początkowa musi zawierać odstępy i musi mieć długość MQ_TRANSACTION_ID_LENGTH. To pole ma zastosowanie tylko wtedy, gdy **sterowanie uow** ma wartość MQCUOWC_FIRST lub MQCUOWC_ONLY.

UOW, element sterujący
UOW. Wartością początkową jest MQCUOWC_ONLY. Możliwe są następujące wartości dodatkowe:

- MQCUOWC_FIRST
- MQCUOWC_MIDDLE,
- MQCUOWC_LAST
- TYLKO MQCUOWC_ONLY
- MQCUOWC_COMMIT
- MQCUOWC_BACKOUT,
- MQCUOWC_CONTINUE

wersja
Numer wersji MQCIH. Wartość początkowa to MQCIH_VERSION_2. Jedyną obsługiwaną wartością jest MQCIH_VERSION_1.

Konstruktory

ImqCICSBridgeHeader();
Konstruktor domyślny.

ImqCICSBridgeHeader(const ImqCICSBridgeHeader & ***nagłówek*** **);**
Konstruktor kopii.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut(ImqMessage & ***komunikat*** **);**
Wstawia strukturę danych MQCIH do buforu komunikatów na początku, przenosząc istniejące dane komunikatu dalej i ustawiając format komunikatu na MQFMT_CICS.

Więcej szczegółów zawiera opis metody klasy macierzystej.

virtual ImqBoolean pasteIn(ImqMessage & ***komunikat*** **);**
Odczytuje strukturę danych MQCIH z buforu komunikatów. Aby operacja zakończyła się pomyślnie, obiekt *msg* musi mieć kodowanie MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT do MQENC_NATIVE. Aby operacja zakończyła się pomyślnie, parametr ImqMessage musi mieć format MQFMT_CICS.

Więcej szczegółów zawiera opis metody klasy macierzystej.

Metody obiektów (publiczne)

void operator = (const ImqCICSBridgeHeader & ***nagłówek*** **);**
Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

MQLONG ADSDescriptor () konst;
Zwraca kopię **deskryptora ADS**.

void setADSDescriptor(const MQLONG ***deskryptor*** **= MQCADSD_NONE);**
Ustawia **deskryptor ADS**.

ImqString attentionIdentifier() konst;
Zwraca kopię **identyfikatora uwagi**dopełnianego odstępami końcowymi do wartości MQ_ATTENTION_ID_LENGTH.

void setAttentionIdentyfikator (const char * *dane* = 0);
Ustawia **identyfikator uwagi**dopełniony odstępami końcowymi na długość MQ_ATTENTION_ID_LENGTH. Jeśli nie podano *danych* , resetuje **identyfikator uwagi** do wartości początkowej.

ImqString authenticator () konst;
Zwraca kopię **elementu uwierzytelniającego**, uzupełnioną odstępami końcowymi do długości MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator(const char * *dane* = 0);
Ustawia **element uwierzytelniający**, dopełniony odstępami końcowymi, na długość MQ_AUTHENTICATOR_LENGTH. Jeśli nie podano *danych* , resetuje **element uwierzytelniający** do wartości początkowej.

ImqString bridgeAbendKod () konst;
Zwraca kopię **kodu nieprawidłowego zakończenia mostu**dopełnianego odstępami końcowymi do długości MQ_ABEND_CODE_LENGTH.

ImqString bridgeCancelKod () konst;
Zwraca kopię **kodu anulowania mostu**dopełnianego odstępami końcowymi do długości MQ_CANCEL_CODE_LENGTH.

void setBridgeCancelCode(const char * *data* = 0);
Ustawia **kod anulowania mostu**, dopełniony odstępami końcowymi, na długość MQ_CANCEL_CODE_LENGTH. Jeśli nie podano *danych* , resetuje **kod anulowania mostu** do wartości początkowej.

Kod bridgeCompletionMQLONG () konst;
Zwraca kopię **kodu zakończenia mostu**.

MQLONG bridgeErrorOffset () konst;
Zwraca kopię **przesunięcia błędu mostu**.

Kod bridgeReasonMQLONG () konst;
Zwraca kopię **kodu przyczyny mostu**.

MQLONG bridgeReturnCode () konst;
Zwraca **kod powrotu mostu**.

conversationalTask() MQLONG konst;
Zwraca kopię **zadania konwersacyjnego**.

void setConversationalTask (const MQLONG *task* = MQCCT_NO);
Ustawia **zadanie konwersacyjne**.

MQLONG cursorPosition() konst;
Zwraca kopię **pozycji kursora**.

void setCursorPosition (const MQLONG *position* = 0);
Ustawia **pozycję kursora**.

MQLONG facilityKeepfacilityKeep () konst;
Zwraca kopię **czasu przechowywania obiektu**.

void setFacilityKeepTime(const MQLONG *czas* = 0);
Ustawia **czas przechowywania narzędzia**.

ImqString facilityLike() konst;
Zwraca kopię narzędzia **, takiego jak**, dopełnione odstępami końcowymi do długości MQ_FACILITY_LIKE_LENGTH.

void setFacilityLike (const char * *nazwa* = 0);
Ustawia narzędzie **, takie jak**, dopełnione odstępami końcowymi, na długość MQ_FACILITY_LIKE_LENGTH. Jeśli *nazwa* nie zostanie podana, resetuje **narzędzie, na przykład** wartość początkową.

ImqBinary facilityToken() konst;
Zwraca kopię **tokenu narzędzia**.

ImqBoolean setFacilityToken (const ImqBinary & *token*);
Ustawia **token narzędzia**. **Długość danych** elementu *token* musi wynosić zero lub MQ_FACILITY_LENGTH. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

void setFacilityToken (const MQBYTE8 *token* = 0);
Ustawia **token narzędzia**. *token* może mieć wartość zero, co jest równoważne określeniu opcji MQCFAC_NONE. Jeśli *token* ma wartość niezerową, musi adresować MQ_FACILITY_LENGTH bajtów danych binarnych. Jeśli używane są wartości predefiniowane, takie jak MQCFAC_NONE, może być konieczne wykonanie rzutowania w celu zapewnienia zgodności sygnatury. Na przykład (MQBYTE *) MQCFAC_NONE.

Funkcja ImqString () konst;
Zwraca kopię **funkcji**dopełnianej odstępami końcowymi do długości MQ_FUNCTION_LENGTH.

Odstęp czasu wywołania MQLONG getWait() konst;
Zwraca kopię **interwału oczekiwania na pobranie**.

void setGetWaitInterval(const MQLONG *interval* = MQCGWI_DEFA
Ustawia **interwał oczekiwania na pobranie**.

MQLONG linkType() konst;
Zwraca kopię **typu powiązania**.

void setLinkType (const MQLONG *type* = MQCLT_PROGRAM);
Ustawia **typ powiązania**.

ImqString nextTransactionIdentyfikator () konst;
Zwraca kopię danych **identyfikatora następnej transakcji** dopełnianą odstępami końcowymi do wartości MQ_TRANSACTION_ID_LENGTH.

MQLONG outputDataLength () konst;
Zwraca kopię **długości danych wyjściowych**.

void setOutputDataLength(const MQLONG *długość* = MQCODL_AS_INPUT);
Ustawia **długość danych wyjściowych**.

ImqString replyToFormat () konst;
Zwraca kopię nazwy w **formacie odpowiedzi** dopełnianej odstępami końcowymi do długości MQ_FORMAT_LENGTH.

void setReplyToFormat(const char * *nazwa* = 0);
Ustawia **format odpowiedzi**, dopełniony odstępami końcowymi, na długość MQ_FORMAT_LENGTH. Jeśli *nazwa* nie zostanie podana, resetuje **format odpowiedzi** do wartości początkowej.

ImqString startCode() konst;
Zwraca kopię **kodu początkowego**z dopełnieniem odstępów końcowych do długości MQ_START_CODE_LENGTH.

void setStartCode (const char * *data* = 0);
Ustawia dane **kodu początkowego** , uzupełnione odstępami końcowymi, na długość MQ_START_CODE_LENGTH. Jeśli nie podano *danych* , resetuje **kod początkowy** do wartości początkowej.

MQLONG taskEnd-Status () konst;
Zwraca kopię **statusu zakończenia zadania**.

ImqString transactionIdentifier() konst;
Zwraca kopię danych **identyfikatora transakcji** dopełnione odstępami końcowymi do długości MQ_TRANSACTION_ID_LENGTH.

void setTransactionIdentyfikator (const char * *dane* = 0);
Ustawia **identyfikator transakcji**dopełniony odstępami końcowymi na długość MQ_TRANSACTION_ID_LENGTH. Jeśli nie podano *danych* , resetuje **identyfikator transakcji** do wartości początkowej.

Funkcja MQLONG UOWControl () konst;
Zwraca kopię **elementu sterującego jednostki pracy**.

void setUOWControl(const MQLONG *control* = MQCUOWC_ONLY);
Ustawia **element sterujący jednostki pracy**.

Wersja MQLONG () konst;
Zwraca numer wersji (**version**).

ImqBoolean setVersion(const MQLONG *wersja* = MQCIH_VERSION_2);
Ustawia numer **version** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Dane obiektu (zabezpieczone)

MQLONG *olVersion*
Maksymalny numer wersji MQCIH, który może być przechowywany w pamięci masowej przydzielonej dla *opcih*.

PMQCIH *opcih*
Adres struktury danych MQCIH. Ilość przydzielonej pamięci masowej jest wskazywana przez *olVersion*.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_WRONG_VERSION

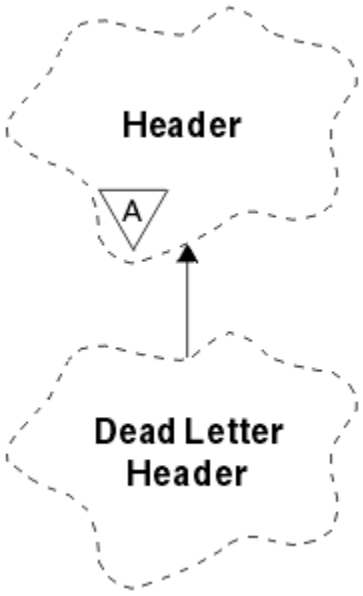
Kody powrotu

Tabela 867. Kody powrotu klasy ImqCICSBridgeHeader

Kod powrotu	Funkcja	CompCode	Przyczyna	Kod nieprawidłowego zakończenia
MQCRC_OK				
BŁĄD MQCRC_BRIDGE_ERROR			MQFB_CICS	
BŁĄD MQCRC_MQ_API_ERROR	IBM MQ call name (nazwa połączenia)	IBM MQ CompCode	IBM MQ Przyczyna	
MQCRC_BRIDGE_TIMEOUT,	IBM MQ call name (nazwa połączenia)	IBM MQ CompCode	IBM MQ Przyczyna	
BŁĄD WYKONYWANIA MQCRC_CICS_EXEC_ERROR	CICS EIBFN,	CICS EIBRESP,	CICS EIBRESP2	
BŁĄD MQCRC_SECURITY_ERROR	CICS EIBFN,	CICS EIBRESP,	CICS EIBRESP2	
MQCRC_PROGRAM_NIEDOSTĘPNE	CICS EIBFN,	CICS EIBRESP,	CICS EIBRESP2	
MQCRC_TRANSID_NIEDOSTĘPNE	CICS EIBFN,	CICS EIBRESP,	CICS EIBRESP2	
MQCRC_BRIDGE_ABEND, nieprawidłowe zakończenie				CICS KOD ABCODE
MQCRC_APPLICATION_ABEND (niepoprawne zakończenie aplikacji MQ)				CICS KOD ABCODE

ImqDeadLetterHeader klasa C + +

Ta klasa obudowuje funkcje struktury danych MQDLH.



Rysunek 19. Klasa ImqDeadLetterHeader

Obiekty tej klasy są zwykle używane przez aplikację, która napotyka komunikat, którego nie można przetworzyć. Nowy komunikat składający się z nagłówka niedostarczonego komunikatu i jego treści jest umieszczany w kolejce niedostarczonych komunikatów, a komunikat jest odrzucany.

- "Atrybuty obiektu" na stronie 1877
- "Konstruktory" na stronie 1878
- "Przeciążone metody ImqItem" na stronie 1878
- "Metody obiektów (publiczne)" na stronie 1878
- "Dane obiektu (zabezpieczone)" na stronie 1879
- "Kody przyczyny" na stronie 1879

Atrybuty obiektu

kod przyczyny niedostarczonych komunikatów
Przyczyna pojawienia się komunikatu w kolejce niedostarczonych komunikatów. Wartością początkową jest MQRC_NONE.

Nazwa menedżera kolejek docelowych
Nazwa oryginalnego docelowego menedżera kolejek. Nazwa jest łańcuchem o długości MQ_Q_MGR_NAME_LENGTH. Jego wartość początkowa wynosi NULL.

nazwa kolejki docelowej
Nazwa oryginalnej kolejki docelowej. Nazwa jest łańcuchem o długości MQ_Q_NAME_LENGTH. Jego wartość początkowa wynosi NULL.

Nazwa aplikacji umieszczającej
Nazwa aplikacji, która wstawiła komunikat do kolejki niedostarczonych komunikatów. Nazwa jest łańcuchem o długości MQ_PUT_APPL_NAME_LENGTH. Jego wartość początkowa wynosi NULL.

Typ aplikacji umieszczającej
Typ aplikacji, która umieściła komunikat w kolejce niedostarczonych komunikatów. Wartością początkową jest zero.

Data umieszczenia
Data wstawienia komunikatu do kolejki niedostarczonych komunikatów. Data jest łańcuchem o długości MQ_PUT_DATE_LENGTH. Jego wartością początkową jest łańcuch pusty.

Czas umieszczenia
Czas wstawienia komunikatu do kolejki niedostarczonych komunikatów. Czas jest łańcuchem o długości MQ_PUT_TIME_LENGTH. Jego wartością początkową jest łańcuch pusty.

Konstruktory

ImqDeadLetterHeader( );
Konstruktor domyślny.

ImqDeadLetterHeader(const ImqDeadLetterHeader & *header*);
Konstruktor kopii.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & *komunikat*);
Wstawia strukturę danych MQDLH do buforu komunikatów na początku, przenosząc istniejące dane komunikatu dalej. Ustawia format *msg* na wartość MQFMT_DEAD_LETTER_HEADER.

Więcej informacji zawiera opis metody klasy ImqHeader na stronie "Klasa C++ ImqHeader" na stronie 1885 .

virtual ImqBoolean pasteIn (ImqMessage & *komunikat*);
Odczytuje strukturę danych MQDLH z buforu komunikatów.

Aby operacja zakończyła się pomyślnie, parametr ImqMessage musi mieć format MQFMT_DEAD_LETTER_HEADER.

Więcej informacji zawiera opis metody klasy ImqHeader na stronie "Klasa C++ ImqHeader" na stronie 1885 .

Metody obiektów (publiczne)

void operator = (const ImqDeadLetterHeader & *nagłówek*);
Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

MQLONG deadLetterReasonCode () konst;
Zwraca kod przyczyny niedostarczonych komunikatów.

void setDeadLetterReasonKod (const MQLONG *przyczyna*);
Ustawia kod przyczyny niedostarczenia.

ImqString destinationQueueManagerName () konst;
Zwraca nazwę docelowego menedżera kolejek bez odstępów końcowych.

void setDestinationQueueManagerNazwa (const char * *nazwa*);
Ustawia nazwę docelowego menedżera kolejek. Obcina dane dłuższe niż MQ_Q_MGR_NAME_LENGTH (48 znaków).

ImqString destinationQueueNazwa () konst;
Zwraca kopię nazwy kolejki docelowej bez odstępów końcowych.

void setDestinationQueueName (const char * *nazwa*);
Ustawia nazwę kolejki docelowej. Obcina dane dłuższe niż MQ_Q_NAME_LENGTH (48 znaków).

ImqString putApplicationNazwa () konst;
Zwraca kopię nazwy aplikacji umieszczającej, bez odstępów końcowych.

void setPutApplicationName (const char * *nazwa* = 0);
Ustawia nazwę aplikacji umieszczającej. Obcina dane dłuższe niż MQ_PUT_APPL_NAME_LENGTH (28 znaków).

MQLONG putApplicationTyp () konst;
Zwraca typ aplikacji umieszczającej.

void setPutApplicationType (const MQLONG *type* = MQAT_NO_CONTEXT);
Ustawia typ aplikacji umieszczającej.

ImqString putDate () konst;
Zwraca kopię daty umieszczenia, bez odstępów końcowych.

void setPutDate (const char * *data* = 0);
Ustawia datę umieszczenia. Obcina dane dłuższe niż MQ_PUT_DATE_LENGTH (8 znaków).

ImqString putTime () konst;
Zwraca kopię czasu umieszczenia, bez odstępów końcowych.

void setPutTime (const char * *time* = 0);
Ustawia czas umieszczenia. Obcina dane dłuższe niż MQ_PUT_TIME_LENGTH (8 znaków).

Dane obiektu (zabezpieczone)

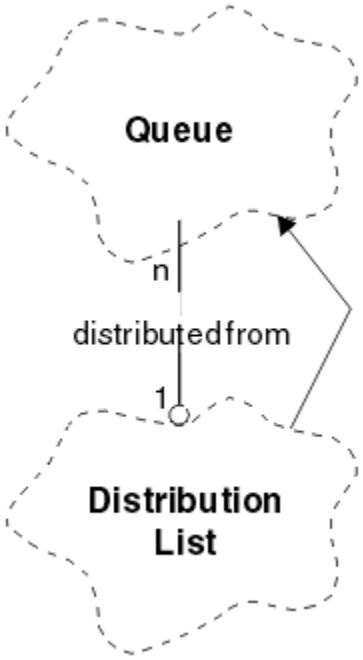
MQDLH *omqdlh*
Struktura danych MQDLH.

Kody przyczyny

- MQRC_INCONSISTENT_FORMAT,
- BŁĄD MQRC_STRUC_ID_ERROR
- BŁĄD MQRC_ENCODING_ERROR

Klasa ImqDistributionList C++

Ta klasa hermetyzuje dynamiczną listę dystrybucyjną, która odwołuje się do jednej lub większej liczby kolejek w celu wysłania komunikatu lub komunikatów do wielu miejsc docelowych.



Rysunek 20. Klasa listy ImqDistribution

- "Atrybuty obiektu" na stronie 1880
- "Konstruktory" na stronie 1880
- "Metody obiektów (publiczne)" na stronie 1880
- "Metody obiektów (zabezpieczone)" na stronie 1880

Atrybuty obiektu

pierwsza kolejka rozproszona
Pierwszy z obiektów klasy, bez określonej kolejności, w którym **lista dystrybucyjna odwołuje się** do tego obiektu.

Początkowo nie ma takich obiektów. Aby pomyślnie otworzyć listę ImqDistribution, musi istnieć co najmniej jeden taki obiekt.

Uwaga: Po otwarciu obiektu listy ImqDistributionwszystkie otwarte obiekty, które się do niego odwołują, są automatycznie zamykane.

Konstruktory

ImqDistributionLista ();
Konstruktor domyślny.

ImqDistributionList (const ImqDistributionList & *list*);
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqDistributionList & *list*);
Wszystkie obiekty, które odwołują się do **tego** obiektu, są wyłuskiwane przed skopiowaniem. Po wywołaniu tej metody żaden obiekt nie będzie się odwoływał do **tego** obiektu.

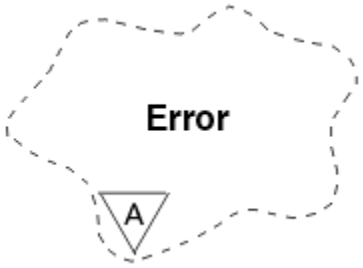
*** firstDistributedQueue () const (konst) ;**
Zwraca **pierwszą kolejkę rozproszoną**.

Metody obiektów (zabezpieczone)

void setFirstDistributedQueue (* *queue* = 0);
Ustawia **pierwszą kolejkę rozproszoną**.

Klasa ImqError C++

Ta klasa abstrakcyjna udostępnia informacje o błędach powiązanych z obiektem.



Rysunek 21. Klasa ImqError

- “Atrybuty obiektu” na stronie 1880
- “Konstruktory” na stronie 1881
- “Metody obiektów (publiczne)” na stronie 1881
- “Metody obiektów (zabezpieczone)” na stronie 1881
- “Kody przyczyny” na stronie 1881

Atrybuty obiektu

kod zakończenia
Najnowszy kod zakończenia. Wartością początkową jest zero. Możliwe są następujące wartości dodatkowe:

- MQCC_OK
- Ostrzeżenie MQCC
- MQCC_FAILED (niepowodzenie MQC)

reason code (kod przyczyny)
Najnowszy kod przyczyny. Wartością początkową jest zero.

Konstruktory

ImqError( );
Konstruktor domyślny.

ImqError(const ImqError & *błąd*);
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqError & *błąd*);
Kopiuje dane instancji z *błąd*, zastępując istniejące dane instancji.

void clearErrorCodes ();
Ustawia zarówno **kod zakończenia** , jak i **kod przyczyny** na zero.

MQLONG completionCode () const (konst) ;
Zwraca **kod zakończenia**.

MQLONG reasonCode () const (konst) ;
Zwraca **kod przyczyny**.

Metody obiektów (zabezpieczone)

ImqBoolean checkReadWskaźnik (const void * *wskaźnik*, const size_t *długość*);
Sprawdza, czy kombinacja wskaźnika i długości jest poprawna dla dostępu tylko do odczytu i zwraca wartość PRAWDA w przypadku powodzenia.

ImqBoolean checkWriteWskaźnik (const void * *wskaźnik*, const size_t *długość*);
Sprawdza, czy kombinacja wskaźnika i długości jest poprawna dla dostępu do odczytu i zapisu i zwraca wartość PRAWDA, jeśli operacja powiedzie się.

void setCompletionCode (const MQLONG *kod* = 0);
Ustawia **kod zakończenia**.

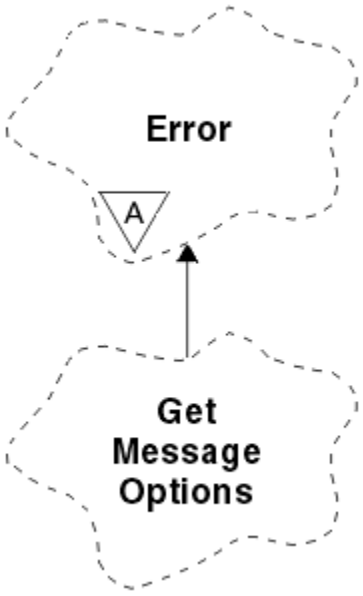
void setReasonCode (const MQLONG *kod* = 0);
Ustawia **kod przyczyny**.

Kody przyczyny

- BŁĄD MQRC_BUFFER_ERROR

Klasa C++ ImqGetMessageOptions

Ta klasa hermetyzuje strukturę danych MQGMO



Rysunek 22. Klasa ImqGetMessageOptions

Atrybuty obiektu

Status grupy
Status komunikatu dla grupy komunikatów. Wartość początkowa to MQGS_NOT_IN_GROUP. Możliwe są następujące wartości dodatkowe:

- MQGS_MSG_IN_GROUP
- MQGS_LAST_MSG_IN_GROUP

opcje dopasowywania
Opcje wyboru komunikatów przychodzących. Wartością początkową jest MQMO_MATCH_MSG_ID | MQMO_MATCH_CORREL_ID. Możliwe są następujące wartości dodatkowe:

- MQMO_GROUP_ID (identyfikator grupy MQ)
- MQMO_MATCH_MSG_SEQ_NUMBER
- MQMO_ZGODNE_PRZESUNIĘCIE
- MQMO_MSG_TOKEN,
- MQMO_BRAK

znacznik komunikatu
Znacznik komunikatu. Wartość binarna (MQBYTE16) o długości MQ_MSG_TOKEN_LENGTH. Wartością początkową jest MQMTOK_NONE.

opcje
Opcje mające zastosowanie do komunikatu. Wartością początkową jest MQGMO_NO_WAIT. Możliwe są następujące wartości dodatkowe:

- 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
- BLOKADA MQGMO_LOCK
- MQGMO_UNLOCK (odblokowanie MQGMO)
- MQGMO_ACCEPT_TRUNCATED_MSG
- MQGMO_SET_SIGNAL
- MQGMO_FAIL_IF_QUIESCING,
- MQGMO_CONVERT
- MQGMO_LOGICAL_ORDER (KOLEJNA_KOLEJNA_STRUKTURA)
- MQGMO_COMPLETE_MSG
- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_NONE

przetłumaczona nazwa kolejki
Nazwa rozstrzygniętej kolejki. Ten atrybut jest tylko do odczytu. Nazwy nigdy nie są dłuższe niż 48 znaków i mogą być dopełniane do tej długości znakami o kodzie zero. Wartością początkową jest łańcuch pusty.

zwrócona długość
Zwrócona długość. Wartością początkową jest MQRL_UNDEFINED. Ten atrybut jest tylko do odczytu.

segmentacja
Możliwość segmentowania komunikatu. Wartością początkową jest MQSEG_INHIBITED. Możliwa jest dodatkowa wartość MQSEG_ALLOWED.

status segmentu
Status segmentacji komunikatu. Wartością początkową jest MQSS_NOT_A_SEGMENT. Możliwe są następujące wartości dodatkowe:

- SEGMENT_MQSS
- MQSS_OSTATNI_SEGMENT

Uczestnictwo w punkcie synchronizacji
Wartość TRUE, jeśli komunikaty są pobierane pod kontrolą punktu synchronizacji.

Interwał oczekiwania
Czas, przez jaki metoda pobierania klasy jest wstrzymywana podczas oczekiwania na nadejście odpowiedniego komunikatu, jeśli taki komunikat nie jest jeszcze dostępny. Wartością początkową jest zero, co ma wpływ na nieokreślony czas oczekiwania. Możliwa jest dodatkowa wartość, MQWI_UNLIMITED. Ten atrybut jest ignorowany, chyba że opcje zawierają opcję MQGMO_WAIT.

Konstruktory

ImqGetMessageOptions( );
Konstruktor domyślny.

ImqGetMessageOptions(const ImqGetMessageOptions & ***gmo*** **);**
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqGetMessageOptions &*gmo*);
Kopiuje dane instancji z komendy *gmo*, zastępując istniejące dane instancji.

MQCHAR groupStatus () konst;
Zwraca status grupy.

void setGroupStatus (const MQCHAR *status*);
Ustawia status grupy.

MQLONG matchOptions () konst;
Zwraca opcje dopasowania.

void setMatchOptions (const MQLONG *opcje*);
Ustawia opcje dopasowania.

ImqBinary messageToken() konst;
Zwraca znacznik komunikatu.

ImqBoolean setMessageToken (const ImqBinary &*token*);
Ustawia znacznik komunikatu. Długość danych elementu *token* musi mieć wartość zero lub wartość MQ_MSG_TOKEN_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

void setMessageToken (const MQBYTE16 *token* = 0);
Ustawia znacznik komunikatu. Parametr *token* może mieć wartość zero, co odpowiada określeniu wartości MQMTOK_NONE. Jeśli atrybut *token* ma wartość różną od zera, musi adresować bajty danych binarnych o długości MQ_MSG_TOKEN_LENGTH.

Jeśli używane są wartości predefiniowane, takie jak MQMTOK_NONE, może nie być konieczne rzutowanie w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQMTOK_NONE.

Opcje MQLONG () konst;
Zwraca opcje.

void setOptions (const MQLONG *opcje*);
Ustawia opcje, w tym wartość udziału w punkcie synchronizacji.

ImqString resolvedQueueNazwa () konst;
Zwraca kopię przetłumaczonych nazw kolejek.

MQLONG returnedLength() konst;
Zwraca zwróconą długość.

Segmentacja MQCHAR () konst;
Zwraca segmentację.

void setSegmentation (const MQCHAR *wartość*);
Ustawia segmentację.

MQCHAR segmentStatus () konst;
Zwraca status segmentu.

void setSegmentStatus (const MQCHAR *status*);
Ustawia status segmentu.

ImqBoolean syncPointUczestnictwo () konst;
Zwraca wartość udziału punktu synchronizacji, która ma wartość TRUE, jeśli opcje obejmują MQGMO_SYNCPOINT lub MQGMO_SYNCPOINT_IF_PERSISTENT.

void setSyncPointParticipation (const ImqBoolean *sync*);
Ustawia wartość udziału w punkcie synchronizacji. Jeśli parametr *sync* ma wartość TRUE, zmienia opcje w celu uwzględnienia MQGMO_SYNCPOINT i wykluczenia zarówno MQGMO_NO_SYNCPOINT, jak i MQGMO_SYNCPOINT_IF_PERSISTENT. Jeśli parametr *sync* ma wartość FALSE, zmienia opcje, aby uwzględnić parametr MQGMO_NO_SYNCPOINT i wykluczyć zarówno parametr MQGMO_SYNCPOINT, jak i parametr MQGMO_SYNCPOINT_IF_PERSISTENT.

MQLONG waitInterval () konst;
Zwraca odstęp czasu oczekiwania.

void setWaitInterval (const MQLONG *interwał*);
Ustawia odstęp czasu oczekiwania.

Metody obiektów (zabezpieczone)

static void setVersionSupported (const MQLONG);
Ustawia wersję MQGMO. Wartością domyślną jest MQGMO_VERSION_3.

Dane obiektu (zabezpieczone)

MQGMO *omqgmo*
Struktura danych MQGMO w wersji 2. Dostęp do pól MQGMO obsługiwanych tylko dla MQGMO_VERSION_2 .

PMQGMO *opgmo*
Adres struktury danych MQGMO. Numer wersji dla tego adresu jest wskazywany w pliku *olVersion*. Sprawdź numer wersji przed uzyskaniem dostępu do pól MQGMO, aby upewnić się, że są one obecne.

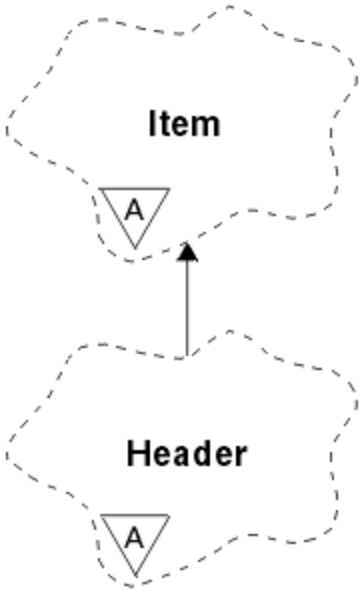
MQLONG *olVersion*
Numer wersji struktury danych MQGMO określonej przez komendę *opgmo*.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

Klasa C++ ImqHeader

Ta klasa abstrakcyjna hermetyzuje wspólne cechy struktury danych MQDLH.



Rysunek 23. Klasa ImqHeader

- "Atrybuty obiektu" na stronie 1885
- "Konstruktory" na stronie 1886
- "Metody obiektów (publiczne)" na stronie 1886

Atrybuty obiektu

zestaw znaków
Oryginalny identyfikator kodowanego zestawu znaków. Początkowo MQCCSI_Q_MGR.

kodowanie
Oryginalne kodowanie. Początkowo MQENC_NATIVE.

format
Oryginalny format. Początkowo MQFMT_NONE.

flagi nagłówka
Wartości początkowe są następujące:

- Zero dla obiektów klasy ImqDeadLetterHeader
- MQIIH_NONE dla obiektów klasy ImqIMSBridgeHeader
- MQRMHF_LAST dla obiektów klasy nagłówka ImqReference
- MQCIH_NONE dla obiektów klasy ImqCICSBridgeHeader
- MQWIH_NONE dla obiektów klasy nagłówka ImqWork

Konstruktory

ImqHeader( );
Konstruktor domyślny.

ImqHeader(const ImqHeader & ***header*** **);**
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqHeader & ***header*** **);**
Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

virtual MQLONG characterSet () const (konst) ;
Zwraca **zestaw znaków**.

virtual void setCharacterSet (const MQLONG ***ccsid*** **= MQCCSI_Q_MGR);**
Ustawia **zestaw znaków**.

wirtualne kodowanie () MQLONG const (konst) ;
Zwraca **kodowanie**.

virtual void setEncoding (const MQLONG ***encoding*** **= MQENC_NATIVE);**
Ustawia **kodowanie**.

virtual ImqString format () const (konst) ;
Zwraca kopię **formatu**, w tym końcowe odstępy.

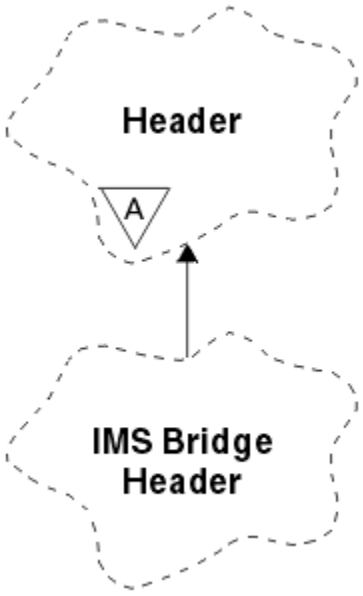
virtual void setFormat (const char * ***nazwa*** **= 0);**
Ustawia **format**dopełniony do 8 znaków odstępami końcowymi.

virtual MQLONG headerFlags () const (konst) ;
Zwraca **flagi nagłówka**.

virtual void setHeaderFlags (const MQLONG ***flags*** **= 0);**
Ustawia **flagi nagłówka**.

ImqIMSBridgeHeader (klasa C + +)

Ta klasa obudowuje funkcje struktury danych MQIIH.



Rysunek 24. Imq, klasaIMSBridgeHeader

Obiekty tej klasy są używane przez aplikacje, które wysyłają komunikaty do mostu IMS przez IBM MQ for z/OS.

Uwaga: Zestaw znaków i kodowanie ImqHeader muszą mieć wartości domyślne i nie mogą być ustawione na inne wartości.

- "Atrybuty obiektu" na stronie 1887
- "Konstruktory" na stronie 1888
- "Przeciążone metody ImqItem" na stronie 1888
- "Metody obiektów (publiczne)" na stronie 1888
- "Dane obiektu (zabezpieczone)" na stronie 1889
- "Kody przyczyny" na stronie 1889

Atrybuty obiektu

element uwierzytelniający
Hasło lub przepustka RACF o długości MQ_AUTHENTICATOR_LENGTH. Wartością początkową jest MQIAUT_NONE.

tryb zatwierdzania
Tryb zatwierdzania. Więcej informacji na temat trybów zatwierdzania IMS zawiera publikacja *OTMA User's Guide* . Wartością początkową jest MQICM_COMMIT_THEN_SEND. Możliwa jest dodatkowa wartość, MQICM_SEND_THEN_COMMIT.

przesłonięcie terminalu logicznego
Przesłonięcie terminalu logicznego o długości MQ_LTERM_OVERRIDE_LENGTH. Wartością początkową jest łańcuch pusty.

nazwa odwzorowania usług formatu komunikatu
Nazwa odwzorowania MFS o długości MQ_MFS_MAP_NAME_LENGTH. Wartością początkową jest łańcuch pusty.

format odpowiedzi
Format dowolnej odpowiedzi o długości MQ_FORMAT_LENGTH. Wartością początkową jest MQFMT_NONE.

zasięg zabezpieczeń
Zasięg przetwarzania zabezpieczeń systemu IMS . Wartością początkową jest MQISS_CHECK. Możliwa jest dodatkowa wartość, MQISS_FULL.

identyfikator instancji transakcji
Tożsamość instancji transakcji, wartość binarna (MQBYTE16) o długości MQ_TRAN_INSTANCE_ID_LENGTH. Wartością początkową jest MQITII_NONE.

Stan transakcji
Stan konwersacji IMS . Wartością początkową jest MQITS_NOT_IN_CONVERSATION. Możliwa jest dodatkowa wartość, MQITS_IN_CONVERSATION.

Konstruktory

ImqIMSBridgeHeader();
Konstruktor domyślny.

ImqIMSBridgeHeader(const ImqIMSBridgeHeader & *nagłówek*);
Konstruktor kopii.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & *komunikat*);
Wstawia strukturę danych MQIIH do buforu komunikatów na początku, przenosząc istniejące dane komunikatów dalej. Ustawia format *msg* na wartość MQFMT_IMS.

Więcej informacji na ten temat zawiera opis metody klasy nadrzędnej.

virtual ImqBoolean pasteIn (ImqMessage & *komunikat*);
Odczytuje strukturę danych MQIIH z buforu komunikatów.

Aby operacja zakończyła się pomyślnie, obiekt *msg* musi mieć kodowanie MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT do MQENC_NATIVE.

Aby działanie było pomyślne, komunikat ImqMessage musi mieć format MQFMT_IMS.

Więcej informacji na ten temat zawiera opis metody klasy nadrzędnej.

Metody obiektów (publiczne)

void operator = (const ImqIMSBridgeHeader & *nagłówek*);
Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

ImqString authenticator () konst;
Zwraca kopię elementu uwierzytelniającego, uzupełnioną odstępami końcowymi do długości MQ_AUTHENTICATOR_LENGTH.

void setAuthenticator (const char * *nazwa*);
Ustawia element uwierzytelniający.

MQCHAR commitMode () konst;
Zwraca tryb zatwierdzania.

void setCommitMode (const MQCHAR *tryb*);
Ustawia tryb zatwierdzania.

ImqString logicalTerminalOverride () konst;
Zwraca kopię nadpisania terminalu logicznego.

void setLogicalTerminalOverride (const char * *override*);
Ustawia nadpisanie terminalu logicznego.

ImqString messageFormatServicesMapNazwa () konst;
Zwraca kopię nazwy odwzorowania usług formatu komunikatu.

void setMessageFormatServicesMapName (const char * *nazwa*);
Ustawia nazwę odwzorowania usług formatu komunikatu.

ImqString replyToFormat () konst;
Zwraca kopię formatu odpowiedzi dopełnione odstępami końcowymi do długości MQ_FORMAT_LENGTH.

void setReplyToFormat (const char * *format*);
Ustawia format odpowiedzi dopełniony odstępami końcowymi do długości MQ_FORMAT_LENGTH.

MQCHAR securityScope () konst;
Zwraca zasięg zabezpieczeń.

void setSecurityScope (const MQCHAR *scope*);
Ustawia zasięg zabezpieczeń.

ImqBinary transactionInstanceId () konst;
Zwraca kopię identyfikatora instancji transakcji.

ImqBoolean setTransactionInstanceId (const ImqBinary & *id*);
Ustawia identyfikator instancji transakcji. Długość danych *tokenu* musi wynosić zero lub wynosić MQ_TRAN_INSTANCE_ID_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

void setTransactionInstanceId (const MQBYTE16 *id* = 0);
Ustawia identyfikator instancji transakcji. *id* może mieć wartość zero, co jest równoważne określeniu wartości MQITII_NONE. Jeśli *id* ma wartość różną od zera, musi adresować bajty danych binarnych o wartości MQ_TRAN_INSTANCE_ID_LENGTH. Jeśli używane są predefiniowane wartości, takie jak MQITII_NONE, konieczne może być wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQITII_NONE.

MQCHAR transactionState () konst;
Zwraca stan transakcji.

void setTransactionState (const MQCHAR *stan*);
Ustawia stan transakcji.

Dane obiektu (zabezpieczone)

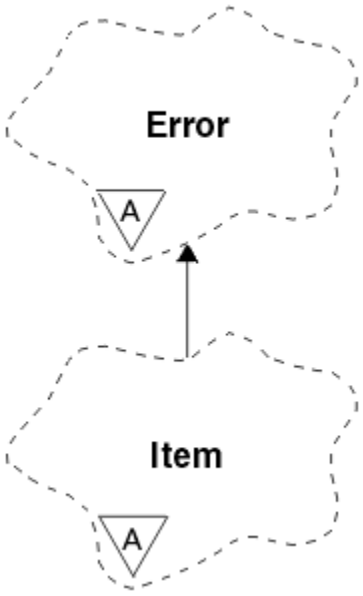
MQIIH *omqiiih*
Struktura danych MQIIH.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_INCONSISTENT_FORMAT,
- BŁĄD MQRC_ENCODING_ERROR
- BŁĄD MQRC_STRUC_ID_ERROR

Klasa ImqItem C++

Ta klasa abstrakcyjna reprezentuje element, być może jeden z kilku, w obrębie komunikatu.



Rysunek 25. Klasa ImqItem

Elementy są konkatenowane razem w buforze komunikatów. Każda specjalizacja jest powiązana z określoną strukturą danych, która rozpoczyna się od identyfikatora struktury.

Metody polimorficzne w tej klasie abstrakcyjnej umożliwiają kopiowanie elementów do i z komunikatów. Metody ImqMessage klasy **readItem** i **writeItem** udostępniają inny styl wywoływania metod polimorficznych, który jest bardziej naturalny dla aplikacji.

- "Atrybuty obiektu" na stronie 1890
- "Konstruktory" na stronie 1890
- "Metody klasy (publiczne)" na stronie 1890
- "Metody obiektów (publiczne)" na stronie 1891
- "Kody przyczyny" na stronie 1891

Atrybuty obiektu

identyfikator struktury
Łańcuch składający się z czterech znaków na początku struktury danych. Ten atrybut jest tylko do odczytu. Ten atrybut należy rozważyć w przypadku klas pochodnych. Nie jest ona dołączana automatycznie.

Konstruktory

ImqItem( );
Konstruktor domyślny.

ImqItem(const ImqItem & ***element*** **);**
Konstruktor kopii.

Metody klasy (publiczne)

static ImqBoolean structureIdIs (const char * ***structure-id-to-test*****, const ImqMessage &** ***msg*** **);**
Zwraca wartość PRAWDA, jeśli **identyfikator struktury** następnego elementu ImqItem w przychodzącym komunikacie *komunikat* jest taki sam, jak *identyfikator struktury do testu*. Następny element jest identyfikowany jako ta część buforu komunikatów, która jest obecnie adresowana przez **wskaźnik danych** ImqCache . Ta metoda opiera się na **identyfikatorze struktury** i dlatego nie ma gwarancji, że będzie działać dla wszystkich klas pochodnych ImqItem .

Metody obiektów (publiczne)

void operator = (const ImqItem & ***element*** **);**
Kopiuje dane instancji z *item*, zastępując istniejące dane instancji.

virtual ImqBoolean copyOut (ImqMessage & ***komunikat*** **) = 0;**
Zapisuje ten obiekt jako następny element w buforze komunikatów wychodzących, dołączając go do istniejących elementów. Jeśli operacja zapisu zakończyła się pomyślnie, zwiększa wartość parametru ImqCache **długość danych**. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Przesłoń tę metodę, aby pracować z konkretną podklasą.

virtual ImqBoolean pasteIn (ImqMessage & ***komunikat*** **) = 0;**
Odczytuje ten obiekt *ze zniszczeniem* z buforu komunikatów przychodzących. Odczyt jest destrukcyjny, ponieważ **wskaźnik danych** ImqCache jest przenoszony dalej. Jednak zawartość buforu pozostaje taka sama, dlatego można ponownie odczytać dane, resetując **wskaźnik danych** ImqCache .

Klasa (sub) tego obiektu musi być spójna z **identyfikatorem struktury** znalezionym obok w buforze komunikatów obiektu *msg* .

Kodowanie obiektu *msg* powinno mieć wartość MQENC_NATIVE. Zaleca się pobieranie komunikatów z ustawionym **kodowaniem** ImqMessage na wartość MQENC_NATIVE i z opcjami **ImqGetMessageOptions** , w tym MQGMO_CONVERT.

Jeśli operacja odczytu powiedzie się, ImqCache **długość danych** zostanie zmniejszona. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

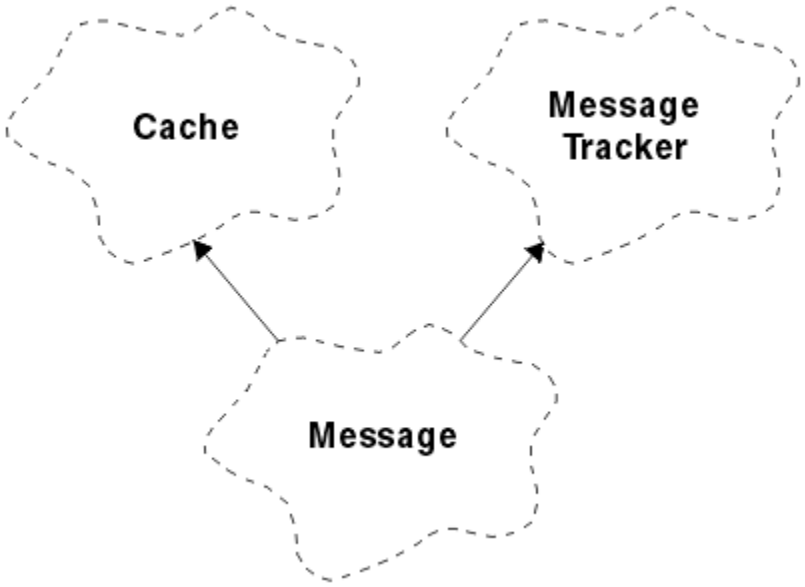
Przesłoń tę metodę, aby pracować z konkretną podklasą.

Kody przyczyny

- BŁĄD MQRC_ENCODING_ERROR
- BŁĄD MQRC_STRUC_ID_ERROR
- MQRC_INCONSISTENT_FORMAT,
- MQRC_INSUFFICIENT_BUFFER (MQRC_INC_BUFFER)
- MQRC_INSUFFICIENT_DATA

Klasa ImqMessage C++

Ta klasa obudowuje strukturę danych MQMD, a także obsługuje konstrukcję i rekonstrukcję danych komunikatu.



Rysunek 26. Klasa ImqMessage

- "Atrybuty obiektu" na stronie 1892
- "Konstruktory" na stronie 1896
- "Metody obiektów (publiczne)" na stronie 1896
- "Metody obiektów (zabezpieczone)" na stronie 1898
- "Dane obiektu (zabezpieczone)" na stronie 1898

Atrybuty obiektu

Dane identyfikatora aplikacji
Informacje o tożsamości powiązane z komunikatem. Wartością początkową jest łańcuch pusty.

Dane źródłowe aplikacji
Informacje o pochodzeniu powiązane z komunikatem. Wartością początkową jest łańcuch pusty.

Liczba wycofań
Liczba sytuacji, w których komunikat został wstępnie pobrany, a następnie wycofany. Wartością początkową jest zero. Ten atrybut jest tylko do odczytu.

zestaw znaków
Identyfikator kodowanego zestawu znaków. Wartością początkową jest MQCCSI_Q_MGR. Możliwe są następujące wartości dodatkowe:

- MQCCSI_INHERIT
- MQCCSI_EMBEDDED

Można również użyć wybranego identyfikatora kodowanego zestawu znaków. Więcej informacji na ten temat zawiera sekcja "konwersja stron kodowych" na stronie 970.

kodowanie
Kodowanie danych komunikatu na komputerze. Wartością początkową jest MQENC_NATIVE.

Koniec ważności
Ilość zależna od czasu, która określa, jak długo program IBM MQ przechowuje niepobrany komunikat przed jego odrzuceniem. Wartością początkową jest MQEI_UNLIMITED.

format
Nazwa formatu (szablonu) opisującego układ danych w buforze. Nazwy dłuższe niż osiem znaków są obcinane do ośmiu znaków. Nazwy są zawsze dopełniane spacjami do ośmiu znaków. Wartością stałej początkowej jest MQFMT_NONE. Możliwe są następujące dodatkowe stałe:

- MQFMT_ADMIN,
- MQFMT_CICS
- MQFMT_COMMAND_1
- MQFMT_COMMAND_2
- MQFMT_DEAD_LETTER_HEADER
- Usługa MQFMT_DIST_HEADER
- MQFMT_EVENT, ZDARZENIE
- MQFMT_IMS,
- MQFMT_IMS_VAR_STRING (łańcuch zmiennych)
- MQFMT_MD_EXTENSION
- MQFMT_PCF,
- MQFMT_REF_MSG_HEADER
- ZMQFMT_RF_HEADER
- MQFMT_STRING
- MQFMT_TRIGGER,
- MQFMT_WORK_INFO_HEADER,
- MQFMT_XMIT_Q_HEADER

Można również użyć wybranego łańcucha specyficznego dla aplikacji. Więcej informacji na ten temat zawiera pole "Format (MQCHAR8) dla deskryptora MQMD" na stronie 459 deskryptora komunikatu (MQMD).

Flagi komunikatów
Informacje sterujące segmentacją. Wartość początkowa to MQMF_SEGMENTATION_INHIBITED. Możliwe są następujące wartości dodatkowe:

- MQMF_SEGMENTATION_ALLOWED
- MQMF_MSG_IN_GROUP
- MQMF_LAST_MSG_IN_GROUP
- MQMF_SEGMENT
- MQMF_LAST_SEGMENT
- MQMF_BRAK

typ komunikatu
Szeroka kategoryzacja komunikatu. Wartością początkową jest MQMT_DATAGRAM. Możliwe są następujące wartości dodatkowe:

- MQMT_SYSTEM_FIRST
- MQMT_SYSTEM_LAST
- MQMT_DATAGRAM,
- MQMT_ŻĄDANIE
- MQMT_REPLY
- MQMT_RAPORT
- MQMT_APPL_FIRST
- MQMT_APPL_LAST

Można również użyć wybranej wartości specyficznej dla aplikacji. Więcej informacji na ten temat zawiera pole "MsgType (MQLONG) w deskryptorze MQMD" na stronie 448 deskryptora komunikatu (MQMD).

Przesunięcie
Informacje o przesunięciu. Wartością początkową jest zero.

Oryginalna długość
Oryginalna długość segmentowanego komunikatu. Wartością początkową jest MQOL_UNDEFINED.

trwałość
Wskazuje, że komunikat jest ważny i musi być zawsze składowany przy użyciu trwałej pamięci masowej. Ta opcja implikuje obniżenie wydajności. Wartością początkową jest MQPER_PERSISTENCE_AS_Q_DEF. Możliwe są następujące wartości dodatkowe:

- MQPER_PERSISTENT
- MQPER_NOT_PERSISTENT

priorytet
Względny priorytet przesyłania i dostarczania. Komunikaty o tym samym priorytecie są zwykle dostarczane w tej samej kolejności, w jakiej zostały dostarczone (chociaż istnieje kilka kryteriów, które muszą być spełnione, aby to zagwarantować). Wartością początkową jest MQPRI_PRIORITY_AS_Q_DEF.

sprawdzanie poprawności właściwości
Określa, czy podczas ustawiania właściwości komunikatu ma być wykonywane sprawdzanie poprawności właściwości. Wartością początkową jest **MQCMHO_DEFAULT_VALIDATION**. Możliwe są następujące wartości dodatkowe:

- MQCMHO_VALIDATE,
- MQCMHO_NO_VALIDATION,

Następujące metody działają w przypadku **sprawdzania poprawności właściwości**:

MQLONG propertyValidation() konst;
Zwraca opcję **sprawdzania poprawności właściwości** .

void setPropertyValidation (const MQLONG *opcja*);
Ustawia opcję **sprawdzania poprawności właściwości** .

Nazwa aplikacji umieszczającej
Nazwa aplikacji, która umieściła komunikat. Wartością początkową jest łańcuch pusty.

Typ aplikacji umieszczającej
Typ aplikacji, która umieściła komunikat. Wartością początkową jest MQAT_NO_CONTEXT. Możliwe są następujące wartości dodatkowe:

- 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_UNKNOWN
- MQAT_USER_FIRST
- Tabela MQAT_USER_LAST

Można również użyć wybranego łańcucha specyficznego dla aplikacji. Więcej informacji na ten temat zawiera pole “PutApplTyp (MQLONG) dla MQMD” na stronie 474 deskryptora komunikatu (MQMD).

Data umieszczenia
Data umieszczenia komunikatu. Wartością początkową jest łańcuch pusty.

Czas umieszczenia
Czas umieszczenia komunikatu. Wartością początkową jest łańcuch pusty.

nazwa menedżera kolejek odpowiedzi
Nazwa menedżera kolejek, do którego ma zostać wysłana dowolna odpowiedź. Wartością początkową jest łańcuch pusty.

nazwa kolejki odpowiedzi
Nazwa kolejki, do której ma zostać wysłana odpowiedź. Wartością początkową jest łańcuch pusty.

raportowanie
Informacje zwrotne powiązane z komunikatem. Wartością początkową jest MQRO_NONE. Możliwe są następujące wartości dodatkowe:

- MQRO_EXCEPTION,
- MQRO_EXCEPTION_WITH_DATA
- MQRO_EXCEPTION_WITH_FULL_DATA *
- MQRO_UTR_WAŻN.
- MQRO_EXPIRATION_WITH_DATA
- MQRO_EXPIRATION_WITH_FULL_DATA *
- MQRO_COA
- MQRO_KOA_WITH_DATA
- MQRO_COA_WITH_FULL_DATA *
- MQRO_COD
- MQRO_KOD_WITH_DATA
- MQRO_KOD_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 (Identyfikator KORELACJI MQ)
- MQRO_DEAD_LETTER_Q
- MQRO_DISCARD_MSG

gdzie * oznacza wartości, które nie są obsługiwane w systemie IBM MQ for z/OS.

numer kolejny
Informacje o sekwencji identyfikujące komunikat w grupie. Wartością początkową jest jeden.

łączna długość komunikatu
Liczba bajtów, które były dostępne podczas ostatniej próby odczytu komunikatu. Ta liczba będzie większa niż ImqCache **długość komunikatu** , jeśli ostatni komunikat został obcięty lub jeśli ostatni komunikat nie został odczytany z powodu obcięcia. Ten atrybut jest tylko do odczytu. Wartością początkową jest zero.

Ten atrybut może być przydatny w każdej sytuacji, w której występują obcięte komunikaty.

ID użytkownika
Tożsamość użytkownika powiązana z komunikatem. Wartością początkową jest łańcuch pusty.

Konstruktory

ImqMessage( );
Konstruktor domyślny.

ImqMessage(const ImqMessage & ***komunikat*** **);**
Konstruktor kopii. Szczegółowe informacje zawiera opis metody **operator =** .

Metody obiektów (publiczne)

void operator = (const ImqMessage & ***komunikat*** **);**
Kopiuje dane MQMD i komunikatu z *msg*. Jeśli użytkownik dostarczył bufor dla tego obiektu, ilość kopiowanych danych jest ograniczona do dostępnej wielkości buforu. W przeciwnym razie system zapewnia udostępnienie buforu o odpowiedniej wielkości dla skopiowanych danych.

ImqString applicationIdDane () const (konst) ;
Zwraca kopię **danych identyfikatora aplikacji**.

void setApplicationIdData (const char * ***data*** **= 0);**
Ustawia **dane identyfikatora aplikacji**.

ImqString applicationOriginData () const (konst) ;
Zwraca kopię **danych pochodzenia aplikacji**.

void setApplicationOriginData (const char * ***data*** **= 0);**
Ustawia **dane źródłowe aplikacji**.

MQLONG backoutCount () const (konst) ;
Zwraca **liczbę wycofań**.

MQLONG characterSet () const (konst) ;
Zwraca **zestaw znaków**.

void setCharacterSet (const MQLONG ***ccsid*** **= MQCCSI_Q_MGR);**
Ustawia **zestaw znaków**.

MQLONG kodowanie () const (konst) ;
Zwraca **kodowanie**.

void setEncoding (const MQLONG ***encoding*** **= MQENC_NATIVE);**
Ustawia **kodowanie**.

MQLONG utrata ważności () const (konst) ;
Zwraca wartość **utrata ważności**.

void setExpiry (const MQLONG ***utrata ważności*** **);**
Ustawia **wygaśnięcie**.

ImqString format () const (konst) ;
Zwraca kopię **formatu**, w tym końcowe odstępy.

ImqBoolean formatIs (const char * ***format-to-test*** **) const (konst) ;**
Zwraca wartość PRAWDA, jeśli **format** jest taki sam jak *format-do-testu*.

void setFormat (const char * ***nazwa*** **= 0);**
Ustawia **format**dopełniony do ośmiu znaków odstępami końcowymi.

MQLONG messageFlags () const (konst) ;
Zwraca **flagi komunikatu**.

void setMessageFlags (const MQLONG ***flagi*** **);**
Ustawia **flagi komunikatu**.

MQLONG messageType () const (konst) ;
Zwraca **typ komunikatu**.

void setMessageType (const MQLONG *typ*);
Ustawia **typ komunikatu**.

MQLONG przesunięcie () const (konst) ;
Zwraca **przesunięcie**.

void setOffset (const MQLONG *przesunięcie*);
Ustawia **przesunięcie**.

MQLONG originalLength () const (konst) ;
Zwraca **pierwotną długość**.

void setOriginalLength (const MQLONG *długość*);
Ustawia **pierwotną długość**.

MQLONG trwałość () const (konst) ;
Zwraca **trwałość**.

void setPersistence (const MQLONG *trwałość*);
Ustawia **trwałość**.

MQLONG priorytet () const (konst) ;
Zwraca **priorytet**.

void setPriority (const MQLONG *priorytet*);
Ustawia **priorytet**.

ImqString putApplicationNazwa () const (konst) ;
Zwraca kopię **nazwy aplikacji umieszczającej**.

void setPutApplicationName (const char * *nazwa* = 0);
Ustawia **nazwę aplikacji umieszczającej**.

MQLONG putApplicationtyp () const (konst) ;
Zwraca **typ aplikacji umieszczającej**.

void setPutApplicationType (const MQLONG *type* = MQAT_NO_CONTEXT);
Ustawia **typ aplikacji umieszczającej**.

ImqString putDate () const (konst) ;
Zwraca kopię **daty umieszczenia**.

void setPutDate (const char * *date* = 0);
Ustawia **datę umieszczenia**.

ImqString putTime () const (konst) ;
Zwraca kopię **czasu umieszczania**.

void setPutTime (const char * *time* = 0);
Ustawia **czas umieszczenia**.

ImqBoolean readItem (ImqItem & *element*);
Odczytuje obiekt *item* z buforu komunikatów za pomocą metody ImqItem **pasteIn** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString replyToQueueManagerNazwa () const (konst) ;
Zwraca kopię **nazwy menedżera kolejek odpowiedzi**.

void setReplyToQueueManagerName (const char * *nazwa* = 0);
Ustawia **nazwę menedżera kolejek odpowiedzi**.

ImqString replyToQueueName () const (konst) ;
Zwraca kopię **nazwy kolejki odpowiedzi**.

void setReplyToQueueName (const char * *nazwa* = 0);
Ustawia **nazwę kolejki odpowiedzi**.

Raport MQLONG () const (konst) ;
Zwraca **raport**.

void setReport (const MQLONG *raport*);
Ustawia **raport**.

MQLONG sequenceNumber () const (konst) ;
Zwraca **numer kolejny**.

void setSequenceNumber (const MQLONG *liczba*);
Ustawia **numer kolejny**.

size_t totalMessageDługość () const (konst) ;
Zwraca **łączną długość komunikatu**.

ImqString userId () const (konst) ;
Zwraca kopię **ID użytkownika**.

void setUserId (const char * *id* = 0);
Ustawia **ID użytkownika**.

ImqBoolean writeItem (ImqItem & *element*);
Zapisuje z obiektu *item* do buforu komunikatów przy użyciu metody ImqItem **copyOut** . Zapis może mieć formę wstawiania, zastępowania lub dołączania: zależy to od klasy obiektu *item* . Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Metody obiektów (zabezpieczone)

static void setVersionSupported (const MQLONG);
Ustawia **wersję MQMD**. Wartością domyślną jest **MQMD_VERSION_2**.

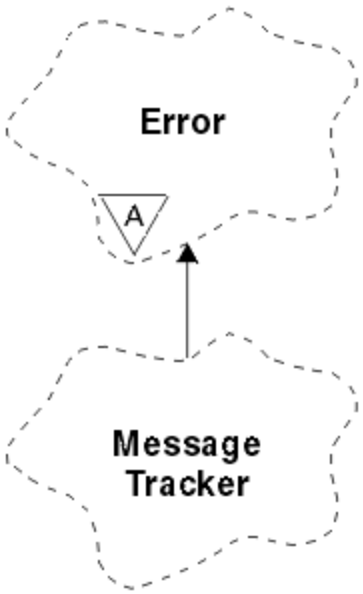
Dane obiektu (zabezpieczone)

z/OS **MQMD1 *omqmd***
Struktura danych MQMD w systemie z/OS.

Multi **MQMD2 *omqmd***
Struktura danych MQMD w systemie Wiele platform.

Klasa ImqMessageTracker C++

Ta klasa hermetyzuje te atrybuty obiektu ImqMessage lub ImqQueue , które mogą być powiązane z dowolnym obiektem.



Rysunek 27. Klasa śledzenia ImqMessage

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji “ImqMessageOdwołanie do śledzenia” na stronie 1847.

Atrybuty obiektu

Token rozliczania
Wartość binarna (MQBYTE32) o długości MQ_ACCOUNTING_TOKEN_LENGTH. Wartością początkową jest MQACT_NONE.

Identyfikator korelacji
Wartość binarna (MQBYTE24) o długości MQ_CORREL_ID_LENGTH, która jest przypisana do korelowania komunikatów. Wartością początkową jest MQCI_NONE. Dodatkowa wartość, MQCI_NEW_SESSION, jest możliwa.

Opinia
Informacje zwrotne, które mają zostać wysłane wraz z komunikatem. Wartością początkową jest MQFB_NONE. Możliwe są następujące wartości dodatkowe:

- MQFB_SYSTEM_FIRST
- MQFB_SYSTEM_LAST
- MQFB_APPL_FIRST
- MQFB_APPL_LAST
- MQFB_COA
- MQFB_COD
- MQFB_UTR_WAŻN.
- MQFB_PAN
- MQFB_NAN
- MQFB_QUIT
- MQFB_DATA_LENGTH_ZERO
- MQFB_DATA_LENGTH_NEGATIVE
- MQFB_DATA_LENGTH_TOO_BIG
- Przepełnienie buforu MQFB_BUFFER_OVERFLOW
- MQFB_LENGTH_OFF_BY_ONE
- MQFB_IIH_BŁĄD
- MQFB_NOT_AUTHORIZED_FOR_IMS (MQFB_NIE_AUTHORITY)
- BŁĄD MQFB_IMS_ERROR
- MQFB_IMS_FIRST
- MQFB_IMS_LAST
- MQFB_CICS_APPL_ABENDED
- MQFB_CICS_APPL_NOT_STARTED (nie uruchomiono)
- MQFB_CICS_BRIDGE_FAILURE
- BŁĄD MQFB_CICS_CCSID_ERROR
- BŁĄD MQFB_CICS_CIH_ERROR
- BŁĄD MQFB_CICS_COMMAREA_ERROR
- BŁĄD MQFB_CICS_CORREL_ID_ERROR
- BŁĄD MQFB_CICS_DLQ_ERROR
- BŁĄD WYWOŁANIA MQFB_CICS_ENCODING_ERROR

- MQFB_CICS_INTERNAL_ERROR (Błąd interfejsu CICS)
- MQFB_CICS_NOT_AUTHORIZED
- MQFB_CICS_UOW_BACKED_OUT
- BŁĄD MQFB_CICS_UOW_ERROR

Można również użyć wybranego łańcucha specyficznego dla aplikacji. Więcej informacji na ten temat zawiera pole "Informacje zwrotne (MQLONG) dla deskryptora MQMD" na stronie 453 deskryptora komunikatu (MQMD).

Identyfikator grupy
Wartość binarna (MQBYTE24) o długości MQ_GROUP_ID_LENGTH, która jest unikalna w obrębie kolejki. Wartością początkową jest MQGI_NONE.

Identyfikator wiadomości
Wartość binarna (MQBYTE24) o długości MQ_MSG_ID_LENGTH, która jest unikalna w obrębie kolejki. Wartością początkową jest MQMI_NONE.

Konstruktory

ImqMessageFunkcja Tracker ();
Konstruktor domyślny.

ImqMessageTracker (const ImqMessageTracker & ***tracker*** **);**
Konstruktor kopii. Szczegółowe informacje zawiera opis metody **operator =** .

Metody obiektów (publiczne)

void operator = (const ImqMessageTracker & ***tracker*** **);**
Kopiuje dane instancji z *programu śledzącego*, zastępując istniejące dane instancji.

ImqBinary accountingToken () const (konst) ;
Zwraca kopię **tokenu rozliczania**.

ImqBoolean setAccountingToken (const ImqBinary & ***token*** **);**
Ustawia **token rozliczania**. **Długość danych** elementu *token* musi wynosić zero lub MQ_ACCOUNTING_TOKEN_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

void setAccountingToken (const MQBYTE32 ***token*** **= 0);**
Ustawia **token rozliczania**. *token* może mieć wartość zero, co jest równoważne określeniu opcji MQACT_NONE. Jeśli *token* ma wartość niezerową, musi adresować MQ_ACCOUNTING_TOKEN_LENGTH bajtów danych binarnych. Jeśli używane są wartości predefiniowane, takie jak MQACT_NONE, konieczne może być wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQACT_NONE.

ImqBinary correlationId () const (konst) ;
Zwraca kopię **identyfikatora korelacji**.

ImqBoolean setCorrelationId (const ImqBinary & ***token*** **);**
Ustawia **identyfikator korelacji**. **Długość danych** elementu *token* musi mieć wartość zero lub wartość MQ_CORREL_ID_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

void setCorrelationId (const MQBYTE24 ***id*** **= 0);**
Ustawia **identyfikator korelacji**. Parametr *id* może mieć wartość zero, co jest równoważne określeniu parametru MQCI_NONE. Jeśli wartość *id* jest różna od zera, musi ona adresować bajty danych binarnych o wartości MQ_CORREL_ID_LENGTH. Jeśli używane są wartości predefiniowane, takie jak MQCI_NONE, konieczne może być wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQCI_NONE.

MQLONG opinia () const (konst) ;
Zwraca **opinię**.

void setFeedback (const MQLONG ***opinia*** **);**
Ustawia **informację zwrotną**.

ImqBinary groupId () const (konst) ;
Zwraca kopię **identyfikatora grupy**.

ImqBoolean setGroupId (const ImqBinary & *token*);
Ustawia **identyfikator grupy**. **Długość danych** elementu *token* musi mieć wartość zero lub wartość MQ_GROUP_ID_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

void setGroupId (const MQBYTE24 *id* = 0);
Ustawia **identyfikator grupy**. Parametr *id* może mieć wartość zero, co jest równoważne określeniu wartości MQGI_NONE. Jeśli parametr *id* ma wartość różną od zera, musi on adresować bajty danych binarnych o długości MQ_GROUP_ID_LENGTH. W przypadku używania predefiniowanych wartości, takich jak MQGI_NONE, może być konieczne wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQGI_NONE.

ImqBinary messageId () const (konst) ;
Zwraca kopię **identyfikatora komunikatu**.

ImqBoolean setMessageId (const ImqBinary & *token*);
Ustawia **identyfikator komunikatu**. **Długość danych** elementu *token* musi mieć wartość zero lub wartość MQ_MSG_ID_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

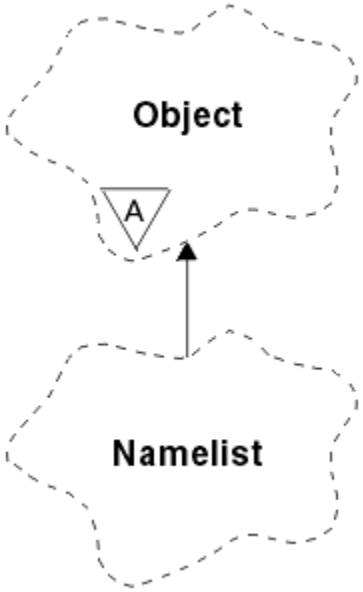
void setMessageId (const MQBYTE24 *id* = 0);
Ustawia **identyfikator komunikatu**. *id* może mieć wartość zero, co jest równoważne określeniu wartości MQMI_NONE. Jeśli wartość *id* jest różna od zera, musi ona adresować bajty danych binarnych o długości MQ_MSG_ID_LENGTH. Jeśli używane są predefiniowane wartości, takie jak MQMI_NONE, konieczne może być wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQMI_NONE.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

Klasa ImqNamelist C++

Ta klasa obudowuje listę nazw.



Rysunek 28. Klasa ImqNamelist

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji “Odwołanie ImqNamelist” na stronie 1847.

- “Atrybuty obiektu” na stronie 1902
- “Konstruktory” na stronie 1902

- “Metody obiektów (publiczne)” na stronie 1902
- “Kody przyczyny” na stronie 1902

Atrybuty obiektu

Liczba nazw
Liczba nazw obiektów w **nazwach list nazw**. Ten atrybut jest tylko do odczytu.

nazwy list nazw
Nazwy obiektów, których liczba jest wskazywana przez **liczbę nazw**. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqNamelist( );
Konstruktor domyślny.

ImqNamelist(const ImqNamelist & ***lista*** **);**
Konstruktor kopii. **Status otwarcia** obiektu ImqObject ma wartość fałsz.

ImqNamelist(const char * ***nazwa*** **);**
Ustawia nazwę ImqObject na **name**.

Metody obiektów (publiczne)

void operator = (const ImqNamelist & ***lista*** **);**
Kopiuje dane instancji z *listy*, zastępując istniejące dane instancji. **Status otwarcia** obiektu ImqObject ma wartość fałsz.

ImqBoolean nameCount(MQLONG & ***liczba*** **);**
Zawiera kopię **name count**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG nameCount ();
Zwraca **liczbę nazw** bez wskazania możliwych błędów.

ImqBoolean namelistName (const MQLONG ***indeks*****, ImqString &** ***nazwa*** **);**
Udostępnia kopię jednej **nazw list nazw** według indeksu zerowego. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

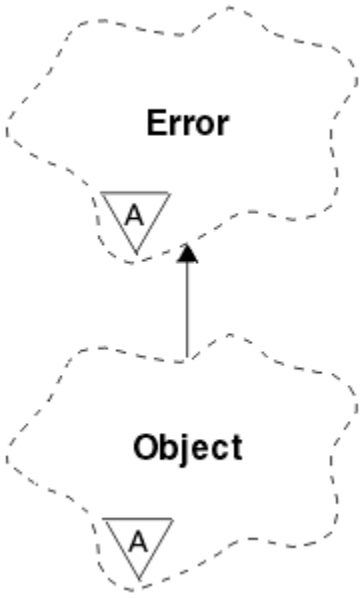
ImqString namelistName (const MQLONG ***indeks*** **);**
Zwraca jedną z **nazw list nazw** według indeksu zerowego bez wskazania możliwych błędów.

Kody przyczyny

- BŁĄD MQRC_INDEX_ERROR
- MQRC_INDEX_NOT_PRESENT (NIEOBECNY)

Klasa C + + ImqObject

Ta klasa jest abstrakcyjna. Po zniszczeniu obiektu tej klasy jest on automatycznie zamykany i zerwane jest jego połączenie z menedżerem ImqQueue.



Rysunek 29. Klasa ImqObject

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji “Odwołanie ImqObject” na stronie 1847.

Atrybuty klasy

zachowanie
Steruje zachowaniem niejawnego otwierania.

IMQ_IMPL_OPEN (8L)
Niejawne otwarcie jest dozwolone. Jest to opcja domyślna.

Atrybuty obiektu

Data zmiany
Data zmiany. Ten atrybut jest tylko do odczytu.

Godzina zmiany
Czas zmiany. Ten atrybut jest tylko do odczytu.

Alternatywny identyfikator użytkownika
Alternatywny identyfikator użytkownika (do znaków MQ_USER_ID_LENGTH). Wartością początkową jest łańcuch pusty.

Alternatywny identyfikator zabezpieczeń
Alternatywny identyfikator zabezpieczeń. Wartość binarna (MQBYTE40) o długości MQ_SECURITY_ID_LENGTH. Wartością początkową jest MQSID_NONE.

opcje zamykania
Opcje, które mają zastosowanie, gdy obiekt jest zamknięty. Wartością początkową jest MQCO_NONE. Ten atrybut jest ignorowany podczas niejawnych operacji ponownego otwierania, w których zawsze używana jest wartość MQCO_NONE.

odwołanie do połączenia
Odwołanie do obiektu menedżera ImqQueue, który udostępnia wymagane połączenie z (lokalnym) menedżerem kolejek. W przypadku obiektu menedżera ImqQueuejest to obiekt sam w sobie. Wartością początkową jest zero.

Uwaga: Nie należy mylić tego elementu z nazwą menedżera kolejek, która identyfikuje menedżer kolejek (prawdopodobnie zdalny) dla nazwanej kolejki.

opis
Opisowa nazwa (do 64 znaków) menedżera kolejek, kolejki, listy nazw lub procesu. Ten atrybut jest tylko do odczytu.

nazwa
Nazwa (maksymalnie 48 znaków) menedżera kolejek, kolejki, listy nazw lub procesu. Wartością początkową jest łańcuch pusty. Nazwa kolejki modelowej zmienia się po **open** na nazwę wynikowej kolejki dynamicznej.

Uwaga: Menedżer ImqQueuemoże mieć pustą nazwę, reprezentującą domyślny menedżer kolejek. Po pomyślnym otwarciu nazwa zostanie zmieniona na rzeczywistą nazwę menedżera kolejek. Lista ImqDistributionjest dynamiczna i musi mieć pustą nazwę.

następny obiekt zarządzany
Jest to następny obiekt tej klasy, bez określonej kolejności, mający takie samo odniesienie do połączenia jak ten obiekt. Wartością początkową jest zero.

Opcje otwarcia
Opcje, które mają zastosowanie po otwarciu obiektu. Wartością początkową jest MQOO_INQUIRE. Istnieją dwa sposoby ustawiania odpowiednich wartości:

1. Nie ustawiaj opcji otwierania i nie używaj metody otwierania. Program IBM MQ automatycznie dopasowuje opcje otwierania oraz automatycznie otwiera, ponownie otwiera i zamyka obiekty zgodnie z potrzebami. Może to spowodować niepotrzebne operacje ponownego otwierania, ponieważ program IBM MQ używa metody openFor , a ta metoda dodaje opcje otwierania tylko przyrostowo.
2. Opcje otwierania należy ustawić przed użyciem metod powodujących wywołanie MQI (patrz sekcja “Odwołanie do interfejsu MQI i C++” na stronie 1840). Dzięki temu niepotrzebne operacje ponownego otwarcia nie będą wykonywane. Należy jawnie ustawić opcje otwierania, jeśli prawdopodobne jest wystąpienie dowolnego z potencjalnych problemów z ponownym otwieraniem (patrz sekcja Ponowne otwieranie).

   Jeśli używana jest metoda open, należy najpierw upewnić się, że opcje open są odpowiednie. Jednak użycie metody open nie jest obowiązkowe. IBM MQ nadal zachowuje się tak samo, jak w przypadku 1, ale w takiej sytuacji zachowanie jest efektywne.

Zero nie jest poprawną wartością; ustaw odpowiednią wartość przed próbą otwarcia obiektu. Można to zrobić za pomocą **setOpenOptions** (*lOpenOptions*), po którym następuje **open** () lub **openFor** (*lRequiredOpenOption*).

Uwaga:

1. Parametr MQOO_OUTPUT jest zastępowany parametrem MQOO_INQUIRE podczas wykonywania metody **open** dla listy dystrybucyjnej, ponieważ parametr MQOO_OUTPUT jest obecnie jedynym poprawnym parametrem **open option** . Jednak dobrą praktyką jest jawne ustawianie opcji MQOO_OUTPUT w aplikacjach, które używają metody **open** .
2. Podaj wartość MQOO_RESOLVE_NAMES, jeśli mają być używane atrybuty **resolved queue manager name** i **resolved queue name** klasy.

status otwarcia
Określa, czy obiekt jest otwarty (TRUE), czy zamknięty (FALSE). Wartością początkową jest FALSE. Ten atrybut jest tylko do odczytu.

poprzedni obiekt zarządzany
Poprzedni obiekt tej klasy, bez określonej kolejności, mający takie samo odniesienie do połączenia jak ten obiekt. Wartością początkową jest zero.

identyfikator-menedżera-kolejek
Identyfikator menedżera kolejek. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqObject( );
Konstruktor domyślny.

ImqObject(const ImqObject & ***obiekt*** **);**
Konstruktor kopii. Status otwarcia będzie miał wartość FAŁSZ.

Metody klasy (publiczne)

statyczne zachowanie MQLONG ();
Zwraca zachowanie.

void setBehavior(const MQLONG ***behavior*** **= 0);**
Ustawia zachowanie.

Metody obiektów (publiczne)

void operator = (const ImqObject & ***obiekt*** **);**
W razie potrzeby wykonuje zamknięcie i kopiuje dane instancji z *obiektu*. Status otwarcia będzie miał wartość FAŁSZ.

ImqBoolean alterationDate(ImqString & ***data*** **);**
Udostępnia kopię daty zmiany. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString alterationDate( );
Zwraca datę modyfikacji bez wskazania możliwych błędów.

ImqBoolean alterationTime(ImqString & ***czas*** **);**
Udostępnia kopię czasu modyfikacji. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString alterationTime( );
Zwraca czas modyfikacji bez wskazania możliwych błędów.

ImqString alternateUserId () konst;
Zwraca kopię alternatywnego identyfikatora użytkownika.

ImqBoolean setAlternateUserId (const char * ***id*** **);**
Ustawia alternatywny identyfikator użytkownika. Alternatywny identyfikator użytkownika można ustawić tylko wtedy, gdy status otwarcia ma wartość FALSE. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBinary alternateSecurityId () konst;
Zwraca kopię alternatywnego identyfikatora zabezpieczeń.

ImqBoolean setAlternateSecurityId(const ImqBinary & ***token*** **);**
Ustawia alternatywny identyfikator zabezpieczeń. Alternatywny identyfikator zabezpieczeń można ustawić tylko wtedy, gdy status otwarcia ma wartość FALSE. Długość danych w polu *token* musi wynosić zero lub wynosić MQ_SECURITY_ID_LENGTH. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean setAlternateSecurityId(const MQBYTE* ***token*** **= 0);**
Ustawia alternatywny identyfikator zabezpieczeń. Parametr *token* może mieć wartość zero, co jest równoważne określeniu parametru MQSID_NONE. Jeśli wartość *token* jest niezerowa, musi adresować bajty danych binarnych o długości MQ_SECURITY_ID_LENGTH. Jeśli używane są wartości

predefiniowane, takie jak MQSID_NONE, konieczne może być wykonanie rzutowania w celu zapewnienia zgodności sygnatur, na przykład (MQBYTE *) MQSID_NONE.

Alternatywny identyfikator zabezpieczeń można ustawić tylko wtedy, gdy status otwarcia ma wartość TRUE. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean setAlternateSecurityId(const unsigned char * *id* = 0);
Ustawia alternatywny identyfikator zabezpieczeń.

ImqBoolean close ();
Ustawia status otwarcia na FALSE. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG closeOptions () konst;
Zwraca opcje zamykania.

void setCloseOptions (const MQLONG *opcje*);
Ustawia opcje zamykania.

ImqQueueManager * connectionReference () konst;
Zwraca odwołanie do połączenia.

void setConnectionReference (ImqQueueManager & *menedżer*);
Ustawia odwołanie do połączenia.

void setConnectionReference (ImqQueueManager * *menedżer* = 0);
Ustawia odwołanie do połączenia.

wirtualny opis ImqBoolean (ImqString & *opis*) = 0;
Udostępnia kopię opisu. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString description ();
Zwraca kopię opisu bez wskazania możliwych błędów.

virtual ImqBoolean name (ImqString & *nazwa*);
Udostępnia kopię nazwy. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString name ();
Zwraca kopię nazwy bez wskazania możliwych błędów.

ImqBoolean setName (const char * *nazwa* = 0);
Ustawia nazwę. Nazwę można ustawić tylko wtedy, gdy status otwarcia ma wartość FALSE, a dla menedżera ImqQueue, gdy status połączenia ma wartość FALSE. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqObject * nextManagedObject () konst;
Zwraca następny obiekt zarządzany.

ImqBoolean open ();
Zmienia status otwarcia na TRUE, otwierając obiekt w razie potrzeby, używając między innymi opcji otwarcia i nazwy. W przypadku tej metody używane są informacje o odwołaniu do połączenia oraz metoda połączenia menedżera ImqQueue, jeśli jest to konieczne, w celu zapewnienia, że status połączenia menedżera ImqQueuema wartość TRUE. Zwraca status otwarcia.

ImqBoolean openFor (const MQLONG *opcje-wymagane* = 0);
Próbuje upewnić się, że obiekt jest otwarty z opcjami otwarcia lub z opcjami otwarcia, które gwarantują zachowanie implikowane przez wartość parametru *required-options* .

Jeśli *required-options* ma wartość zero, dane wejściowe są wymagane i wszystkie opcje wejściowe są wystarczające. Tak więc, jeśli otwarte opcje już zawierają jedną z następujących wartości:

- MQOO_INPUT_AS_Q_DEF
- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE (wyłączna)

Opcje otwarcia są już zadowalające i nie zostały zmienione. Jeśli opcje otwarcia nie zawierają jeszcze żadnej z tych opcji, w opcjach otwarcia jest ustawiona opcja MQOO_INPUT_AS_Q_DEF.

Jeśli opcja *required-options* ma wartość niezerową, wymagane opcje są dodawane do opcji open. Jeśli *required-options* jest dowolną z tych opcji, pozostałe opcje zostaną zresetowane.

Jeśli którakolwiek z opcji otwarcia została zmieniona, a obiekt jest już otwarty, obiekt jest zamykany tymczasowo i ponownie otwierany w celu dostosowania opcji otwarcia.

Zwraca TRUE, jeśli operacja zakończyła się pomyślnie. Powodzenie wskazuje, że obiekt jest otwarty z odpowiednimi opcjami.

MQLONG openOptions () konst;
Zwraca opcje otwarcia.

ImqBoolean setOpenOptions (const MQLONG *opcje*);
Ustawia opcje otwierania. Opcje otwarcia mogą być ustawione tylko wtedy, gdy status otwarcia ma wartość FALSE. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean openStatus () konst;
Zwraca status otwarcia.

ImqObject * previousManagedObject () konst;
Zwraca poprzedni obiekt zarządzany.

ImqBoolean queueManagerIdentyfikator (ImqString & *id*);
Udostępnia kopię identyfikatora menedżera kolejek. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString queueManagerIdentifier ();
Zwraca identyfikator menedżera kolejek bez wskazania możliwych błędów.

Metody obiektów (zabezpieczone)

virtual ImqBoolean closeTemporarily ();
Umożliwia bezpieczne zamknięcie obiektu przed ponownym otwarciem. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie. Ta metoda zakłada, że status otwarcia ma wartość PRAWDA.

MQHCONN connectionHandle () konst;
Zwraca wartość MQHCONN powiązaną z odwołaniem do połączenia. Ta wartość wynosi zero, jeśli nie ma odwołania do połączenia lub jeśli menedżer nie jest połączony.

ImqBoolean inquire (const MQLONG *int-attr*, MQLONG & *wartość*);
Zwraca liczbę całkowitą, której indeks jest wartością MQIA_ *. W przypadku błędu wartość jest ustawiana na MQIAV_UNDEFINED.

ImqBoolean inquire (const MQLONG *atr*, char * & *bufor*, const size_t *długość*);
Zwraca łańcuch znaków, którego indeks jest wartością MQCA_ *.

Uwaga: Obie te metody zwracają tylko jedną wartość atrybutu. Jeśli *obraz stanu* jest wymagany dla więcej niż jednej wartości, a wartości są ze sobą spójne przez chwilę, program IBM MQ C++ nie udostępnia tej funkcji i należy użyć wywołania MQINQ z odpowiednimi parametrami.

virtual void openInformationDisperse ();
Rozprasza informacje z sekcji zmiennych struktury danych MQOD bezpośrednio po wywołaniu MQOPEN.

virtual ImqBoolean openInformationPrepare ();
Przygotowuje informacje dla sekcji zmiennej struktury danych MQOD bezpośrednio przed wywołaniem MQOPEN i zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean set (const MQLONG *int-attr*, const MQLONG *wartość*);
Ustawia IBM MQ atrybut całkowitoliczbowy.

ImqBoolean set (const MQLONG *char-attr*, const char * *buffer*, const size_t *długość_wymagana*);
Ustawia atrybut znakowy IBM MQ .

void setNextManagedObject (const ImqObject * *object* = 0);
Ustawia następny obiekt zarządzany.

Uwaga: tej funkcji należy używać tylko wtedy, gdy jest się pewnym, że nie spowoduje ona złamania listy obiektów zarządzanych.

void setPreviousManagedObject (const ImqObject * *object* = 0);
Ustawia poprzedni obiekt zarządzany.

Uwaga: tej funkcji należy używać tylko wtedy, gdy jest się pewnym, że nie spowoduje ona złamania listy obiektów zarządzanych.

Dane obiektu (zabezpieczone)

MQHOBJ *ohobj*
Uchwyt obiektu IBM MQ (poprawny tylko wtedy, gdy status otwarcia ma wartość TRUE).

MQOD *omqod*
Wbudowana struktura danych MQOD. Ilość pamięci masowej przydzielonej dla tej struktury danych jest wymagana dla programu MQOD w wersji 2. Sprawdź numer wersji *(omqod.Version)* i uzyskaj dostęp do innych pól w następujący sposób:

MQOD_VERSION_1
Można uzyskać dostęp do wszystkich innych pól w *omqod* .

MQOD_VERSION_2
Można uzyskać dostęp do wszystkich innych pól w *omqod* .

MQOD_VERSION_3
omqod.pmqod jest wskaźnikiem do dynamicznie przydzielanego, większego MQOD. Nie można uzyskać dostępu do innych pól w *omqod* . Dostęp do wszystkich pól adresowanych przez plik *omqod.pmqod* jest możliwy.

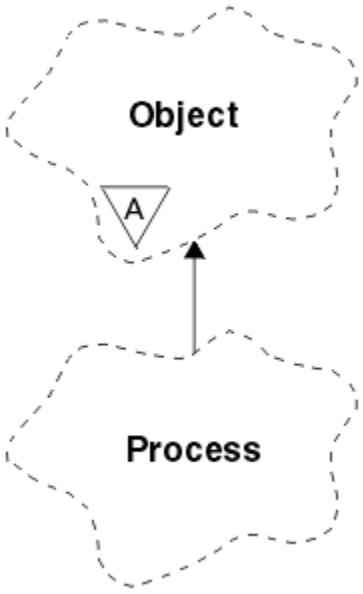
Uwaga: *omqod.pmqod.Version* może być mniejsza niż *omqod.Version*, co oznacza, że IBM MQ MQI client ma więcej funkcji niż serwer IBM MQ .

Kody przyczyny

- MQRC_ATTRIBUTE_LOCKED (atrybut mqrc_locked)
- MQRC_INCONSISTENT_OBJECT_STATE,
- MQRC_NO_CONNECTION_REFERENCE,
- MQRC_STORAGE_NIEDOSTĘPNY
- MQRC_REOPEN_SAVED_CONTEXT_ERR
- (kody przyczyny z MQCLOSE)
- (kody przyczyny z MQCONN)
- (kody przyczyny z MQINQ)
- (kody przyczyny z MQOPEN)
- (kody przyczyny z MQSET)

Klasa ImqProcess C++

Ta klasa hermetyzuje proces aplikacji (obiekt IBM MQ typu MQOT_PROCESS), który może być wyzwalany przez monitor wyzwalacza.



Rysunek 30. Klasa ImqProcess

- "Atrybuty obiektu" na stronie 1909
- "Konstruktory" na stronie 1909
- "Metody obiektów (publiczne)" na stronie 1909

Atrybuty obiektu

Identyfikator aplikacji
Tożsamość procesu aplikacji. Ten atrybut jest tylko do odczytu.

Typ aplikacji
Typ procesu aplikacji. Ten atrybut jest tylko do odczytu.

Dane środowiska
Informacje o środowisku dla procesu. Ten atrybut jest tylko do odczytu.

Dane użytkownika
Dane użytkownika dla procesu. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqProcess( );
Konstruktor domyślny.

ImqProcess(const ImqProcess & *proces*);
Konstruktor kopii. Parametr ImqObject **status otwarcia** ma wartość FALSE.

ImqProcess(const char * *nazwa*);
Ustawia **nazwę**obiektu ImqObject .

Metody obiektów (publiczne)

void operator = (const ImqProcess & *proces*);
W razie potrzeby wykonuje operację zamykania, a następnie kopiuje dane instancji z *procesu*. Opcja ImqObject **status otwarcia** będzie mieć wartość FALSE.

ImqBoolean applicationId (ImqString & *id*);
Udostępnia kopię **identyfikatora aplikacji**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString applicationId ( );
Zwraca **identyfikator aplikacji** bez wskazania możliwych błędów.

ImqBoolean applicationType (MQLONG & *typ*);
Udostępnia kopię **typu aplikacji**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG applicationType ();
Zwraca **typ aplikacji** bez wskazania możliwych błędów.

ImqBoolean environmentData (ImqString & *dane*);
Udostępnia kopię **danych środowiska**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

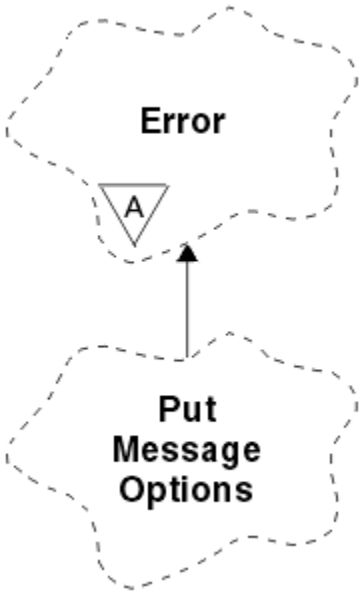
ImqString environmentData ( );
Zwraca **dane środowiska** bez wskazania możliwych błędów.

ImqBoolean userData (ImqString & *dane*);
Udostępnia kopię **danych użytkownika**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString userData ( );
Zwraca **dane użytkownika** bez wskazania możliwych błędów.

ImqPutMessageOptions klasa C++

Ta klasa hermetyzuje strukturę danych MQPMO.



Rysunek 31. Klasa ImqPutMessageOptions

- "Atrybuty obiektu" na stronie 1910
- "Konstruktory" na stronie 1911
- "Metody obiektów (publiczne)" na stronie 1911
- "Dane obiektu (zabezpieczone)" na stronie 1912
- "Kody przyczyny" na stronie 1912

Atrybuty obiektu

odwołanie do kontekstu
ImqQueue , która udostępnia kontekst dla komunikatów. Początkowo nie ma odwołania.

opcje
Opcje umieszczania komunikatów. Wartością początkową jest MQPMO_NONE. Możliwe są następujące wartości dodatkowe:

- 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_UPRAWNIENIA_UŻYTKOWNIKA
- MQPMO_FAIL_IF_QUIESCING,

pola rekordu
Flagi, które sterują dołączaniem rekordów umieszczania komunikatów podczas umieszczania komunikatu. Wartością początkową jest MQPMRF_NONE. Możliwe są następujące wartości dodatkowe:

- MQPMRF_ID_komunikatu
- MQPMRF_CORREL_ID (Identyfikator relacji MQ)
- MQPMRF_GROUP_ID (identyfikator grupy MQPMRF_ID)
- MQPMRF_FEEDBACK
- MQPMRF_ACCOUNTING_TOKEN,

ImqMessageAtrybuty śledzenia są pobierane z obiektu dla każdego określonego pola. ImqMessageAtrybuty śledzenia są pobierane z obiektu ImqMessage dla każdego pola, które nie jest określone.

rozstrzygnięta nazwa menedżera kolejek
Nazwa docelowego menedżera kolejek określona podczas umieszczania. Wartość początkowa jest pusta. Ten atrybut jest tylko do odczytu.

przetłumaczona nazwa kolejki
Nazwa kolejki docelowej określona podczas umieszczania. Wartość początkowa jest pusta. Ten atrybut jest tylko do odczytu.

Uczestnictwo w punkcie synchronizacji
Wartość TRUE, gdy komunikaty są umieszczane pod kontrolą punktu synchronizacji.

Konstruktory

ImqPutMessageOptions( );
Konstruktor domyślny.

ImqPutMessageOptions(const ImqPutMessageOptions & ***pmo*** **);**
Konstruktor kopii.

Metody obiektów (publiczne)

void operator = (const ImqPutMessageOptions & ***pmo*** **);**
Kopiuje dane instancji z komendy *pmo*, zastępując istniejące dane instancji.

ImqQueue * contextReference () konst;
Zwraca odwołanie do kontekstu.

void setContextReference (const ImqQueue & ***kolejka*** **);**
Ustawia odwołanie do kontekstu.

void setContextReference (const ImqQueue * ***queue*** **= 0);**
Ustawia odwołanie do kontekstu.

Opcje MQLONG () konst;
Zwraca opcje.

void setOptions (const MQLONG *opcje*);
Ustawia opcje, w tym wartość udziału w punkcie synchronizacji.

MQLONG recordFields () konst;
Zwraca pola rekordu.

void setRecordFields (const MQLONG *fields*);
Ustawia pola rekordu.

ImqString resolvedQueueManagerName () konst;
Zwraca kopię rozstrzygniętej nazwy menedżera kolejek.

ImqString resolvedQueueNazwa () konst;
Zwraca kopię przetłumaczonych nazw kolejek.

ImqBoolean syncPointUczestnictwo () konst;
Zwraca wartość udziału punktu synchronizacji, która ma wartość TRUE, jeśli opcje zawierają parametr MQPMO_SYNCPOINT.

void setSyncPointParticipation (const ImqBoolean *sync*);
Ustawia wartość udziału w punkcie synchronizacji. Jeśli parametr *sync* ma wartość TRUE, opcje są zmieniane w celu uwzględnienia parametru MQPMO_SYNCPOINT i wykluczenia parametru MQPMO_NO_SYNCPOINT. Jeśli parametr *sync* ma wartość FALSE, opcje są zmieniane w celu uwzględnienia opcji MQPMO_NO_SYNCPOINT i wykluczenia opcji MQPMO_SYNCPOINT.

Dane obiektu (zabezpieczone)

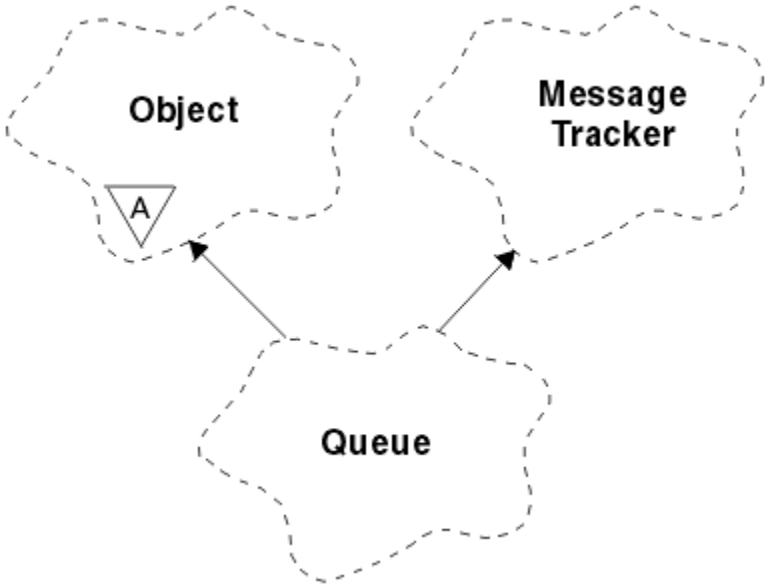
MQPMO *omqpmo*
Struktura danych MQPMO.

Kody przyczyny

- MQRC_STORAGE_NIEDOSTĘPNY

Klasa ImqQueue C++

Ta klasa hermetyzuje kolejkę komunikatów (obiekt IBM MQ typu MQOT_Q).



Rysunek 32. Klasa ImqQueue

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji Tabela 862 na stronie 1849.

Atrybuty obiektu

nazwa kolejki wycofanych komunikatów
Nadmierna liczba wycofanych nazw. Ten atrybut jest tylko do odczytu.

Próg wycofania
Próg wycofania. Ten atrybut jest tylko do odczytu.

nazwa kolejki podstawowej
Nazwa kolejki, na którą alias jest tłumaczony. Ten atrybut jest tylko do odczytu.

nazwa klastra
Nazwa klastra. Ten atrybut jest tylko do odczytu.

Nazwa listy nazw klastrów
Nazwa listy nazw klastrów. Ten atrybut jest tylko do odczytu.

Klasyfikacja obciążenia klastrów
Stopień obciążenia klastra. Ten atrybut jest tylko do odczytu.

Priorytet obciążenia klastrów
Priorytet obciążenia klastra. Ten atrybut jest tylko do odczytu.

Kolejka użycia obciążenia klastra
Obciążenie klastra używa wartości kolejki. Ten atrybut jest tylko do odczytu.

data utworzenia
Dane tworzenia kolejki. Ten atrybut jest tylko do odczytu.

Czas utworzenia
Czas utworzenia kolejki. Ten atrybut jest tylko do odczytu.

Bieżące zapełnienie
Liczba komunikatów w kolejce. Ten atrybut jest tylko do odczytu.

powiązanie domyślne
Domyślne powiązanie. Ten atrybut jest tylko do odczytu.

Domyślna opcja otwarcia wejścia
Domyślna opcja open-for-input. Ten atrybut jest tylko do odczytu.

Trwałość domyślna
Domyślna trwałość komunikatu. Ten atrybut jest tylko do odczytu.

Domyślny priorytet
Domyślny priorytet komunikatu. Ten atrybut jest tylko do odczytu.

Typ definicji
Typ definicji kolejki. Ten atrybut jest tylko do odczytu.

Zdarzenie o dużej głębokości
Atrybut sterujący dla zdarzeń dużego zapełnienia kolejki. Ten atrybut jest tylko do odczytu.

górny limit głębokości
Górny limit głębokości kolejki. Ten atrybut jest tylko do odczytu.

zdarzenie niskiego poziomu głębokości
Atrybut sterujący dla zdarzeń niskiego zapełnienia kolejki. Ten atrybut jest tylko do odczytu.

dolny limit głębokości
Dolny limit głębokości kolejki. Ten atrybut jest tylko do odczytu.

maksymalna głębokość zdarzenia
Atrybut sterujący dla zdarzeń maksymalnej głębokości kolejki. Ten atrybut jest tylko do odczytu.

odwołanie do listy dystrybucyjnej
Opcjonalne odwołanie do listy ImqDistribution, której można użyć do dystrybucji komunikatów do więcej niż jednej kolejki, w tym do tej kolejki. Wartość początkowa jest pusta.

Uwaga: Po otwarciu obiektu ImqQueue każdy otwarty obiekt listy ImqDistribution, do którego się odwołuje, jest automatycznie zamykany.

listy dystrybucyjne
Możliwość obsługi list dystrybucyjnych przez kolejkę transmisji. Ten atrybut jest tylko do odczytu.

nazwa kolejki dynamicznej
Nazwa kolejki dynamicznej. Wartością początkową jest AMQ.* dla wszystkich platform AIX, Linux, and Windows .

Zapisane wycofane komunikaty
Informacja o tym, czy należy zatwardzić liczbę wycofań. Ten atrybut jest tylko do odczytu.

Typ indeksu
Typ indeksu. Ten atrybut jest tylko do odczytu.

Nie zezwalaj na pobieranie
Określa, czy operacje pobierania są dozwolone. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko dla kolejki aliasowej lub kolejki lokalnej.

Zablokuj umieszczenie
Określa, czy operacje umieszczania (put) są dozwolone. Wartość początkowa zależy od definicji kolejki.

Nazwa kolejki inicjacji
Nazwa kolejki inicjującej. Ten atrybut jest tylko do odczytu.

Maksymalna głębokość
Maksymalna liczba komunikatów dozwolonych w kolejce. Ten atrybut jest tylko do odczytu.

Maksymalna długość komunikatu
Maksymalna długość dowolnego komunikatu w tej kolejce, która może być mniejsza niż maksymalna długość dla dowolnej kolejki zarządzanej przez powiązany menedżer kolejek. Ten atrybut jest tylko do odczytu.

Kolejność dostarczania komunikatów
Określa, czy priorytet komunikatu jest istotny. Ten atrybut jest tylko do odczytu.

następna kolejka rozproszona
Następny obiekt tej klasy, bez określonej kolejności, mający takie samo **odwołanie do listy dystrybucyjnej** jak ten obiekt. Wartością początkową jest zero.

Jeśli obiekt w łańcuchu zostanie usunięty, poprzedni obiekt i następny obiekt zostaną zaktualizowane, tak aby ich dowiązania do kolejki rozproszonej nie wskazywały już na usunięty obiekt.

nietrwała klasa komunikatu
Poziom niezawodności dla nietrwałych komunikatów umieszczanych w tej kolejce. Ten atrybut jest tylko do odczytu.

Liczba otwartych wejść
Liczba obiektów ImqQueue otwartych do wprowadzania. Ten atrybut jest tylko do odczytu.

Liczba otwartych wyjść
Liczba obiektów ImqQueue otwartych do wyprowadzania. Ten atrybut jest tylko do odczytu.

poprzednia kolejka rozproszona
Poprzedni obiekt tej klasy, bez określonej kolejności, mający takie samo **odwołanie do listy dystrybucyjnej** jak ten obiekt. Wartością początkową jest zero.

Jeśli obiekt w łańcuchu zostanie usunięty, poprzedni obiekt i następny obiekt zostaną zaktualizowane, tak aby ich dowiązania do kolejki rozproszonej nie wskazywały już na usunięty obiekt.

Nazwa procesu
Nazwa definicji procesu. Ten atrybut jest tylko do odczytu.

Rozliczanie kolejek
Poziom informacji rozliczeniowych dla kolejek. Ten atrybut jest tylko do odczytu.

nazwa_menedżera_kolejek
Nazwa menedżera kolejek (prawdopodobnie zdalnego), w którym znajduje się kolejka. Nie należy mylić menedżera kolejek o podanej tutaj nazwie z **odwołaniem do połączenia** ImqObject , które odwołuje się do (lokalnego) menedżera kolejek udostępniającego połączenie. Wartość początkowa jest pusta.

Monitorowanie kolejek
Poziom gromadzenia danych monitorowania dla kolejki. Ten atrybut jest tylko do odczytu.

Statystyka kolejek
Poziom danych statystycznych dla kolejki. Ten atrybut jest tylko do odczytu.

Typ kolejki
Typ kolejki. Ten atrybut jest tylko do odczytu.

Nazwa zdalnego menedżera kolejek
Nazwa zdalnego menedżera kolejek. Ten atrybut jest tylko do odczytu.

Nazwa zdalnej kolejki
Nazwa zdalnej kolejki znana w zdalnym menedżerze kolejek. Ten atrybut jest tylko do odczytu.

rozstrzygnięta nazwa menedżera kolejek
Rozstrzygnięta nazwa menedżera kolejek. Ten atrybut jest tylko do odczytu.

przetłumaczona nazwa kolejki
Nazwa rozstrzygniętej kolejki. Ten atrybut jest tylko do odczytu.

Interwał przechowywania
Odstęp czasu przechowywania kolejki. Ten atrybut jest tylko do odczytu.

zasięg
Zasięg definicji kolejki. Ten atrybut jest tylko do odczytu.

interwał usług
Odstęp czasu usługi. Ten atrybut jest tylko do odczytu.

zdarzenie interwału usług
Atrybut sterujący dla zdarzeń odstępu czasu usługi. Ten atrybut jest tylko do odczytu.

Możliwość współużytkowania
Określa, czy kolejka może być współużytkowana. Ten atrybut jest tylko do odczytu.

klasa pamięci masowej
Klasa pamięci. Ten atrybut jest tylko do odczytu.

Nazwa kolejki transmisji
Nazwa kolejki przesyłania. Ten atrybut jest tylko do odczytu.

Kontrola wyzwalacza
Element sterujący wyzwalacza. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko dla kolejki lokalnej.

Dane wyzwalacza
Dane wyzwalacza. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko dla kolejki lokalnej.

Wyzwalacz uruchamiany zapełnieniem
Wyzwalacz uruchamiany zapełnieniem. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko dla kolejki lokalnej.

Priorytet komunikatu wyzwalacza
Priorytet komunikatu progowego dla wyzwalaczy. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko dla kolejki lokalnej.

typ wyzwalacza
Typ wyzwalacza. Wartość początkowa zależy od definicji kolejki. Ten atrybut jest poprawny tylko dla kolejki lokalnej.

wykorzystanie
Użycie. Ten atrybut jest tylko do odczytu.

Konstruktory

ImqQueue( );
Konstruktor domyślny.

ImqQueue(const ImqQueue & ***kolejka*** **);**
Konstruktor kopii. Opcja ImqObject **status otwarcia** będzie mieć wartość FALSE.

ImqQueue(const char * ***nazwa*** **);**
Ustawia **nazwę**obiektu ImqObject .

Metody obiektów (publiczne)

void operator = (const ImqQueue & ***queue*** **);**
W razie potrzeby wykonuje operację zamykania, a następnie kopiuje dane instancji z *kolejki*. Opcja ImqObject **status otwarcia** będzie mieć wartość FALSE.

ImqBoolean backoutRequeueNazwa (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy wycofanej kolejki**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString backoutRequeueNazwa ();
Zwraca **nazwę kolejki wycofanych komunikatów** bez wskazania możliwych błędów.

ImqBoolean backoutThreshold (MQLONG & ***próg*** **);**
Zawiera kopię **progu wycofania**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG backoutThreshold ();
Zwraca wartość **progu wycofania** bez wskazania możliwych błędów.

ImqBoolean baseQueueNazwa (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy kolejki podstawowej**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString baseQueueNazwa ();
Zwraca **nazwę kolejki podstawowej** bez wskazania możliwych błędów.

ImqBoolean clusterName(ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy klastra**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString clusterName( );
Zwraca **nazwę klastra** bez wskazania możliwych błędów.

ImqBoolean clusterNamelistNazwa (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy listy nazw klastrów**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString clusterNamelistName ();
Zwraca **nazwę listy nazw klastrów** bez wskazania błędów.

ImqBoolean clusterWorkLoadPriority (MQLONG & priority);
Udostępnia kopię wartości priorytetu obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterWorkLoadPriority ();
Zwraca wartość priorytetu obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkLoadRank (MQLONG & rank);
Udostępnia kopię wartości klasyfikacji obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterWorkLoadRank ();
Zwraca wartość klasyfikacji obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Udostępnia kopię wartości kolejki użycia obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterWorkLoadUseQ ();
Zwraca wartość kolejki użycia obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean creationDate (ImqString & *data*);
Zawiera kopię **daty utworzenia**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString creationDate ( );
Zwraca **datę utworzenia** bez wskazania możliwych błędów.

ImqBoolean creationTime (ImqString & *czas*);
Udostępnia kopię **czasu utworzenia**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString creationTime ( );
Zwraca **czas utworzenia** bez wskazania możliwych błędów.

ImqBoolean currentDepth (MQLONG & *głębokość*);
Udostępnia kopię **bieżącej głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG currentDepth ();
Zwraca **bieżącą głębokość** bez wskazania możliwych błędów.

ImqBoolean defaultInputOpenOption (MQLONG & *opcja*);
Udostępnia kopię **domyślnej opcji otwierania wejścia**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG defaultInputOpenOption ();
Zwraca **domyślną opcję otwarcia wejścia** bez wskazania możliwych błędów.

ImqBoolean defaultPersistence (MQLONG & *trwałość*);
Udostępnia kopię **domyślnej trwałości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG defaultPersistence ();
Zwraca **trwałość domyślną** bez wskazania możliwych błędów.

ImqBoolean defaultPriority (MQLONG & *priorytet*);
Udostępnia kopię **domyślnego priorytetu**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG defaultPriority ();
Zwraca **priorytet domyślny** bez wskazania możliwych błędów.

ImqBoolean defaultBind (MQLONG & *powiązanie*);
Udostępnia kopię **powiązania domyślnego**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG defaultBind ();
Zwraca **domyślne powiązanie** bez wskazania możliwych błędów.

ImqBoolean definitionType (MQLONG & *typ*);
Udostępnia kopię **typu definicji**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG definitionType ();
Zwraca **typ definicji** bez wskazania możliwych błędów.

ImqBoolean depthHighZdarzenie (MQLONG & *zdarzenie*);
Udostępnia kopię stanu włączenia **zdarzenia o dużej głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG depthHighZdarzenie ();
Zwraca stan włączenia **zdarzenia głębokości dużej** bez wskazania możliwych błędów.

ImqBoolean depthHighLimit (MQLONG & *limit*);
Udostępnia kopię **górnego limitu głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG depthHighlimit ();
Zwraca wartość **górnego limitu głębokości** bez wskazania możliwych błędów.

ImqBoolean depthLowZdarzenie (MQLONG & *zdarzenie*);
Udostępnia kopię stanu włączenia **zdarzenia niskiego poziomu głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG depthLowZdarzenie ();
Zwraca stan włączenia **zdarzenia niskiego poziomu głębokości** bez wskazania możliwych błędów.

ImqBoolean depthLowLimit (MQLONG & *limit*);
Udostępnia kopię **dolnego limitu głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG depthLowlimit ();
Zwraca wartość **dolnego limitu głębokości** bez wskazania możliwych błędów.

ImqBoolean depthMaximumEvent (MQLONG & ***zdarzenie*** **);**
Udostępnia kopię stanu włączenia **zdarzenia maksymalnej głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG depthMaximumEvent ();
Zwraca stan włączenia **zdarzenia maksymalnej głębokości** bez wskazania możliwych błędów.

ImqDistributionList * distributionListSkorowidz () const (konst) ;
Zwraca **odwołanie do listy dystrybucyjnej**.

void setDistributionListReference (ImqDistributionList & ***lista*** **);**
Ustawia **odwołanie do listy dystrybucyjnej**.

void setDistributionListReference (ImqDistributionList * ***list*** **= 0);**
Ustawia **odwołanie do listy dystrybucyjnej**.

ImqBoolean distributionLists (MQLONG & ***obsługa*** **);**
Udostępnia kopię wartości **list dystrybucyjnych** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG distributionLists ();
Zwraca wartość **list dystrybucyjnych** bez wskazania możliwych błędów.

ImqBoolean setDistributionLists (const MQLONG ***obsługa*** **);**
Ustawia wartość **list dystrybucyjnych** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString dynamicQueueNazwa () const (konst) ;
Zwraca kopię **nazwy kolejki dynamicznej**.

ImqBoolean setDynamicQueueName (const char * ***nazwa*** **);**
Ustawia **nazwę kolejki dynamicznej**. **Nazwa kolejki dynamicznej** można ustawić tylko wtedy, gdy właściwość ImqObject **status otwarcia** ma wartość FALSE. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean pobierz (ImqMessage & ***komunikat*****, ImqGetMessageOptions &** ***opcje*** **);**
Pobiera komunikat z kolejki przy użyciu określonych *opcji*. W razie potrzeby wywołuje metodę ImqObject **openFor** , aby upewnić się, że **opcje otwarcia** ImqObject zawierają jedną z wartości MQOO_INPUT_ * lub wartość MQOO_BROWSE, w zależności od *opcji*. Jeśli obiekt *msg* ma **automatyczny bufor** ImqCache , bufor ten zwiększa się w celu uwzględnienia pobranych komunikatów. Metoda **clearMessage** jest wywoływana dla obiektu *msg* przed pobraniem.

Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Uwaga: Wynikiem wywołania metody jest wartość FALSE, jeśli **kod przyczyny** ImqObject ma wartość MQRC_TRUNCATED_MSG_FAILED, nawet jeśli ten **kod przyczyny** został sklasyfikowany jako ostrzeżenie. Jeśli obcięty komunikat jest akceptowany, **długość komunikatu** ImqCache odzwierciedla obciętą długość. W obu przypadkach wartość ImqMessage **łączna długość komunikatu** wskazuje liczbę dostępnych bajtów.

ImqBoolean pobierz (ImqMessage & ***komunikat*** **);**
Podobnie jak w przypadku poprzedniej metody, z tą różnicą, że używane są domyślne opcje pobierania komunikatów.

ImqBoolean get (ImqMessage & ***msg*****, ImqGetMessageOptions &** ***opcje*****, const size_t** ***wielkość_buforu*** **);**
Podobnie jak w przypadku dwóch poprzednich metod, z tą różnicą, że wskazywana jest przesłaniająca wartość *buffer-size* . Jeśli obiekt *msg* korzysta z **automatycznego buforu** ImqCache , metoda **resizeBuffer** jest wywoływana dla obiektu *msg* przed pobraniem komunikatu, a bufor nie powiększa się, aby pomieścić większy komunikat.

ImqBoolean get (ImqMessage & ***msg*****, const size_t** ***wielkość_buforu*** **);**
Podobnie jak w przypadku poprzedniej metody, z tą różnicą, że używane są domyślne opcje pobierania komunikatów.

ImqBoolean hardenGetBackout (MQLONG &*utwardzanie*);
Udostępnia kopię wartości **harden get backout** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG hardenGetBackout ();
Zwraca wartość **harden get backout** bez wskazania możliwych błędów.

ImqBoolean indexType(MQLONG &*typ*);
Udostępnia kopię **typu indeksu**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG indexType();
Zwraca **typ indeksu** bez wskazania możliwych błędów.

ImqBoolean inhibitGet (MQLONG &*zablokuj*);
Udostępnia kopię wartości **inhibit get** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG inhibitGet ();
Zwraca wartość **inhibit get** bez wskazania możliwych błędów.

ImqBoolean setInhibitGet (const MQLONG *zablokuj*);
Ustawia wartość **inhibit get** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean inhibitPut (MQLONG &*zablokuj*);
Udostępnia kopię wartości **inhibit put** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG inhibitPut ();
Zwraca wartość **inhibit put** bez wskazania możliwych błędów.

ImqBoolean setInhibitPut (const MQLONG *zablokuj*);
Ustawia wartość **inhibit put** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean initiationQueue (ImqString &*nazwa*);
Udostępnia kopię **nazwy kolejki inicjującej**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString initiationQueueNazwa ();
Zwraca **nazwę kolejki inicjującej** bez wskazania możliwych błędów.

ImqBoolean maximumDepth (MQLONG &*głębokość*);
Udostępnia kopię **maksymalnej głębokości**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumDepth ();
Zwraca **maksymalną głębokość** bez wskazania możliwych błędów.

ImqBoolean maximumMessageDługość (MQLONG &*długość*);
Udostępnia kopię **maksymalnej długości komunikatu**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumMessagedługość ();
Zwraca **maksymalną długość komunikatu** bez wskazania możliwych błędów.

ImqBoolean messageDeliverysekwencja (MQLONG &*sekwencja*);
Udostępnia kopię **sekwencji dostarczania komunikatów**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG messageDeliverySequence ();
Zwraca wartość **sekwencji dostarczania komunikatów** bez wskazania możliwych błędów.

ImqQueue * nextDistributedKolejka () const (konst) ;
Zwraca **następną kolejkę rozproszoną**.

ImqBoolean nonPersistentMessageClass (MQLONG & monq);
Udostępnia kopię wartości klasy nietrwałego komunikatu. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG nonPersistentMessageClass ();
Zwraca wartość klasy nietrwałego komunikatu bez wskazania możliwych błędów.

ImqBoolean openInputopenInput (MQLONG &*liczba*);
Udostępnia kopię **liczby otwartych danych wejściowych**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG openInputliczba ();
Zwraca **liczbę otwartych danych wejściowych** bez wskazania możliwych błędów.

ImqBoolean openOutputliczba (MQLONG & ***liczba*** **);**
Zawiera kopię **liczby otwartych danych wyjściowych**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG openOutputopenOutput ();
Zwraca **liczbę otwartych danych wyjściowych** bez wskazania możliwych błędów.

ImqQueue * previousDistributedKolejka () const (konst) ;
Zwraca **poprzednią kolejkę rozproszoną**.

ImqBoolean processName (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy procesu**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString processName ( );
Zwraca **nazwę procesu** bez wskazania możliwych błędów.

ImqBoolean put (ImqMessage & ***komunikat*** **);**
Umieszcza komunikat w kolejce przy użyciu domyślnych opcji umieszczania komunikatu. W razie potrzeby używa metody ImqObject **openFor** , aby upewnić się, że **opcje otwarcia** ImqObject zawierają opcję MQOO_OUTPUT.

Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean put (ImqMessage & ***komunikat*****, ImqPutMessageOptions &** ***pmo*** **);**
Umieszcza komunikat w kolejce przy użyciu określonej komendy *pmo*. W razie potrzeby używa metody ImqObject **openFor** , aby upewnić się, że produkt ImqObject **opcje otwierania** zawiera parametr MQOO_OUTPUT i (jeśli *pmo* **opcje** zawiera dowolną z wartości MQPMO_PASS_IDENTITY_CONTEXT, MQPMO_PASS_ALL_CONTEXT, MQPMO_SET_IDENTITY_CONTEXT lub MQPMO_SET_ALL_CONTEXT) odpowiadających wartościom MQOO_ * _CONTEXT.

Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Uwaga: Jeśli komenda *pmo* zawiera **odwołanie do kontekstu**, obiekt, do którego istnieje odwołanie, jest otwierany, jeśli jest to konieczne, w celu udostępnienia kontekstu.

ImqBoolean queueAccounting (MQLONG & acctq);
Udostępnia kopię wartości rozliczania kolejki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueAccounting ();
Zwraca wartość rozliczania kolejki bez wskazania możliwych błędów.

ImqString queueManagerNazwa () const (konst) ;
Zwraca **nazwę menedżera kolejek**.

ImqBoolean setQueueManagerName (const char * ***nazwa*** **);**
Ustawia **nazwę menedżera kolejek**. **Nazwę menedżera kolejek** można ustawić tylko wtedy, gdy właściwość ImqObject **status otwarcia** ma wartość FALSE. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean queueMonitoring (MQLONG & monq);
Udostępnia kopię wartości monitorowania kolejki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueMonitoring ();
Zwraca wartość monitorowania kolejki bez wskazania możliwych błędów.

ImqBoolean queueStatistics (MQLONG & statq);
Udostępnia kopię wartości statystyki kolejki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueStatistics ();
Zwraca wartość statystyki kolejki bez wskazania możliwych błędów.

ImqBoolean queueType (MQLONG & ***typ*** **);**
Udostępnia kopię wartości **typu kolejki** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueType ();
Zwraca **typ kolejki** bez wskazania możliwych błędów.

ImqBoolean remoteQueueManagerName (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy zdalnego menedżera kolejek**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString remoteQueueManagerName ( );
Zwraca **nazwę zdalnego menedżera kolejek** bez wskazania możliwych błędów.

ImqBoolean remoteQueueName (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy kolejki zdalnej**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString remoteQueueNazwa ();
Zwraca **nazwę kolejki zdalnej** bez wskazania możliwych błędów.

ImqBoolean resolvedQueueManagerName(ImqString & ***nazwa*** **);**
Udostępnia kopię **rozstrzygniętej nazwy menedżera kolejek**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Uwaga: Ta metoda kończy się niepowodzeniem, chyba że parametr MQOO_RESOLVE_NAMES jest wśród **opcji otwierania** ImqObject .

ImqString resolvedQueueManagerName( ) ;
Zwraca **rozstrzygniętą nazwę menedżera kolejek**bez wskazania możliwych błędów.

ImqBoolean resolvedQueueNazwa (ImqString & ***nazwa*** **);**
Udostępnia kopię **rozstrzygniętej nazwy kolejki**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Uwaga: Ta metoda kończy się niepowodzeniem, chyba że parametr MQOO_RESOLVE_NAMES jest wśród **opcji otwierania** ImqObject .

ImqString resolvedQueueNazwa ();
Zwraca **rozstrzygniętą nazwę kolejki**bez wskazania możliwych błędów.

ImqBoolean retentionInterval (MQLONG & ***przedział_czasu*** **);**
Udostępnia kopię **przedziału czasu przechowywania**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG retentionInterval ();
Zwraca **przedział czasu przechowywania** bez wskazania możliwych błędów.

ImqBoolean zasięg (MQLONG & ***zasięg*** **);**
Udostępnia kopię **zasięgu**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG zasięg ();
Zwraca **zasięg** bez wskazania możliwych błędów.

ImqBoolean serviceInterval (MQLONG & ***odstęp czasu*** **);**
Zawiera kopię **przedziału czasu usługi**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG serviceInterval ();
Zwraca **przedział czasu usługi** bez wskazania możliwych błędów.

ImqBoolean serviceIntervalEvent (MQLONG & ***zdarzenie*** **);**
Udostępnia kopię stanu włączenia **zdarzenia odstępu czasu usługi**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG serviceIntervalEvent ();
Zwraca stan włączenia **zdarzenia odstępu czasu usługi** bez wskazania możliwych błędów.

ImqBoolean współużytkowalność (MQLONG & ***współużytkowalność*** **);**
Udostępnia kopię wartości **shareability** (współużytkowalność). Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG współużytkowalność ();
Zwraca wartość **współużytkowalności** bez wskazania możliwych błędów.

ImqBoolean storageClass(ImqString & ***klasa*** **);**
Udostępnia kopię **klasy pamięci masowej**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString storageClass( );
Zwraca **klasę pamięci masowej** bez wskazania możliwych błędów.

ImqBoolean transmissionQueueNazwa (ImqString & ***nazwa*** **);**
Udostępnia kopię **nazwy kolejki transmisji**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString transmissionQueueNazwa ();
Zwraca **nazwę kolejki transmisji** bez wskazania możliwych błędów.

ImqBoolean triggerControl (MQLONG & ***element sterujący*** **);**
Udostępnia kopię wartości **elementu sterującego wyzwalacza** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG triggerControl ();
Zwraca wartość **elementu sterującego wyzwalacza** bez wskazania możliwych błędów.

ImqBoolean setTriggerControl (const MQLONG ***element sterujący*** **);**
Ustawia wartość **elementu sterującego wyzwalacza** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean triggerData (ImqString & ***dane*** **);**
Udostępnia kopię **danych wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString triggerData ( );
Zwraca kopię **danych wyzwalacza** bez wskazania możliwych błędów.

ImqBoolean setTriggerDane (const char * ***dane*** **);**
Ustawia **dane wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean triggerDepth (MQLONG & ***głębokość*** **);**
Udostępnia kopię **głębokości wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG triggerDepth ();
Zwraca **głębokość wyzwalacza** bez wskazania możliwych błędów.

ImqBoolean setTriggerDepth (const MQLONG ***głębokość*** **);**
Ustawia **głębokość wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean triggerMessagePriorytet (MQLONG & ***priorytet*** **);**
Udostępnia kopię **priorytetu komunikatu wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG triggerMessagepriorytet ();
Zwraca **priorytet komunikatu wyzwalacza** bez wskazania możliwych błędów.

ImqBoolean setTriggerMessagePriority (const MQLONG ***priorytet*** **);**
Ustawia **priorytet komunikatu wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean triggerType (MQLONG & ***typ*** **);**
Udostępnia kopię **typu wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG triggerType ();
Zwraca **typ wyzwalacza** bez wskazania możliwych błędów.

ImqBoolean setTriggerType (const MQLONG ***typ*** **);**
Ustawia **typ wyzwalacza**. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean składnia (MQLONG & ***składnia*** **);**
Udostępnia kopię wartości **usage** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG składnia ();
Zwraca wartość **usage** bez wskazania możliwych błędów.

Metody obiektów (zabezpieczone)

void setNextDistributedQueue (ImqQueue * ***queue*** **= 0);**
Ustawia **następną kolejkę rozproszoną**.

Uwaga: Funkcji tej należy używać tylko wtedy, gdy jest się pewnym, że nie spowoduje ona przerwania listy kolejek rozproszonych.

void setPreviousDistributedQueue (ImqQueue * ***queue*** **= 0);**
Ustawia **poprzednią kolejkę rozproszoną**.

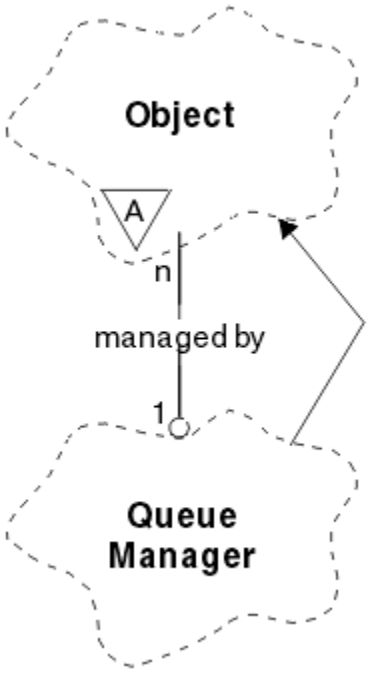
Uwaga: Funkcji tej należy używać tylko wtedy, gdy jest się pewnym, że nie spowoduje ona przerwania listy kolejek rozproszonych.

Kody przyczyny

- MQRC_ATTRIBUTE_LOCKED (atrybut mqrc_locked)
- MQRC_CONTEXT_OBJECT_NOT_VALID
- BŁĄD MQRC_CONTEXT_OPEN_ERROR
- MQRC_CURSOR_NOT_VALID (MQRC_CURSOR_NOT_VALID
- MQRC_NO_BUFFER
- BŁĄD MQRC_REOPEN_EXCL_INPUT_ERROR
- BŁĄD MQRC_REOPEN_INQUIRE_ERROR
- BŁĄD MQRC_REOPEN_TEMPORARY_Q_ERROR
- (kody przyczyny z MQGET)
- (kody przyczyny z MQPUT)

Klasa ImqQueueManager C++

Ta klasa hermetyzuje menedżer kolejek (obiekt IBM MQ typu MQOT_Q_MGR).



Rysunek 33. Klasa menedżera ImqQueue

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji “Odwołanie do menedżera ImqQueue” na stronie 1851. Nie wszystkie wymienione metody mają zastosowanie do wszystkich platform. Więcej informacji na ten temat zawiera sekcja ALTER QMGR .

- “Atrybuty klasy” na stronie 1924
- “Atrybuty obiektu” na stronie 1924
- “Konstruktory” na stronie 1930
- “Destruktory” na stronie 1930
- “Metody klasy (publiczne)” na stronie 1930
- “Metody obiektów (publiczne)” na stronie 1930

- “Metody obiektów (zabezpieczone)” na stronie 1939
- “Dane obiektu (zabezpieczone)” na stronie 1939
- “Kody przyczyny” na stronie 1939

Atrybuty klasy

zachowanie
Steruje zachowaniem niejawnego połączenia i rozłączenia.

IMQ_00_DISC_BACKOUT (0L)
Jawne wywołanie metody disconnect oznacza wycofanie. Ten atrybut wyklucza się wzajemnie z IMQ_EXPL_DISC_COMMIT.

IMQ_EXPL_DISC_COMMIT (1L)
Jawne wywołanie metody disconnect oznacza zatwierdzenie (wartość domyślna). Ten atrybut wyklucza się wzajemnie z atrybutem IMQ_EXPL_DISC_BACKOUT.

IMQ_IMPL_CONN (2L)
Połączenie niejawne jest dozwolone (wartość domyślna).

IMQ_IMPL_DISC_BACKOUT (0L)
Niejawne wywołanie metody disconnect, które może wystąpić podczas niszczenia obiektu, oznacza wycofanie. Ten atrybut wyklucza się wzajemnie z parametrem IMQ_IMPL_DISC_COMMIT.

IMQ_IMPL_DISC_COMMIT (4L)
Niejawne wywołanie metody disconnect, które może wystąpić podczas niszczenia obiektu, oznacza zatwierdzenie (domyślnie). Ten atrybut wyklucza się wzajemnie z wartością IMQ_IMPL_DISC_BACKOUT.

W produkcie IBM MQ V7.0 lub nowszej aplikacje C + +, które korzystają z niejawnego połączenia, muszą określać parametr IMQ_IMPL_CONN wraz z innymi opcjami udostępnionymi w metodzie `setBehavior()` dla obiektu klasy `ImqQueueManager`. Jeśli aplikacja nie używa metody `setBehavior()` do jawnego ustawienia opcji zachowania, na przykład:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

Ta zmiana nie ma wpływu, ponieważ parametr MQ_IMPL_CONN jest domyślnie włączony.

Jeśli aplikacja jawnie ustawia opcje zachowania, na przykład:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_DISC_COMMIT)
```

Należy dołączyć parametr IMQ_IMPL_CONN w metodzie `setBehavior()` w następujący sposób, aby umożliwić aplikacji nawiązanie niejawnego połączenia:

```
ImqQueueManager_object.setBehavior(IMQ_IMPL_CONN | IMQ_IMPL_DISC_COMMIT)
```

Atrybuty obiektu

przesłonięcie połączeń rozliczeniowych
Umożliwia aplikacjom przesłanianie ustawień rozliczania MQI i rozliczania kolejek values.This jest tylko do odczytu.

Interwał rozliczania
Czas przed zapisaniem pośrednich rekordów rozliczeniowych (w sekundach). Ten atrybut jest tylko do odczytu.

Zapis aktywności
Steruje generowaniem raportów działania. Ten atrybut jest tylko do odczytu.

Sprawdzenie dołączenia nowego MCA
Elementy sprawdzane w celu określenia, czy agent MCA powinien zostać adoptowany w przypadku wykrycia nowego kanału przychodzącego, który ma taką samą nazwę jak agent MCA, który jest już aktywny. Ten atrybut jest tylko do odczytu.

Typ dołączenia nowego MCA
Określa, czy osierocona instancja agenta MCA określonego typu kanału powinna zostać automatycznie zrestartowana po wykryciu nowego żądania kanału przychodzącego zgodnego z parametrami sprawdzania adoptowania nowego agenta MCA. Ten atrybut jest tylko do odczytu.

Typ uwierzytelniania
Wskazuje typ wykonywanego uwierzytelniania.

zdarzenie uprawnień
Steruje zdarzeniami uprawnień. Ten atrybut jest tylko do odczytu.

opcje początku
Opcje mające zastosowanie do metody begin. Wartością początkową jest MQBO_NONE.

zdarzenie mostu
Określa, czy są generowane zdarzenia mostu IMS . Ten atrybut jest tylko do odczytu.

Automatyczna definicja kanału
Wartość automatycznej definicji kanału. Ten atrybut jest tylko do odczytu.

zdarzenie automatycznej definicji kanału
Wartość zdarzenia automatycznej definicji kanału. Ten atrybut jest tylko do odczytu.

Wyjście automatycznej definicji kanału
Nazwa wyjścia automatycznej definicji kanału. Ten atrybut jest tylko do odczytu.

zdarzenie kanału
Określa, czy generowane są zdarzenia kanału. Ten atrybut jest tylko do odczytu.

Adaptery inicjatora kanału
Liczba podzadań adaptera, które mają być używane do przetwarzania wywołań IBM MQ . Ten atrybut jest tylko do odczytu.

Kontrola inicjatora kanału
Określa, czy inicjator kanału ma być uruchamiany automatycznie podczas uruchamiania menedżera kolejek. Ten atrybut jest tylko do odczytu.

Programy rozsyłające inicjatora kanału
Liczba programów rozsyłających, które mają być używane dla inicjatora kanału. Ten atrybut jest tylko do odczytu.

autostart śledzenia inicjatora kanału
Określa, czy śledzenie inicjatora kanału ma być uruchamiane automatycznie, czy nie. Ten atrybut jest tylko do odczytu.

Wielkość tabeli śledzenia inicjatora kanału
Wielkość obszaru danych śledzenia inicjatora kanału (w MB). Ten atrybut jest tylko do odczytu.

Monitorowanie kanałów
Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kanałów. Ten atrybut jest tylko do odczytu.

odwołanie do kanału
Odwołanie do definicji kanału do użycia podczas połączenia klienta. Po nawiązaniu połączenia ten atrybut można ustawić na wartość NULL, ale nie można go zmienić na inną wartość. Wartość początkowa jest pusta.

Statystyka kanałów
Steruje kolekcjonowaniem danych statystycznych dla kanałów. Ten atrybut jest tylko do odczytu.

zestaw znaków
Identyfikator kodowanego zestawu znaków (CCSID). Ten atrybut jest tylko do odczytu.

Monitorowanie nadawcy klastrów
Steruje gromadzeniem danych monitorowania bezpośredniego dla automatycznie definiowanych kanałów nadawczych klastra. Ten atrybut jest tylko do odczytu.

Statystyka nadawcy klastrów
Steruje gromadzeniem danych statystycznych dla automatycznie zdefiniowanych kanałów nadajnika klastra. Ten atrybut jest tylko do odczytu.

Dane obciążenia klastra
Dane wyjścia obciążenia klastra. Ten atrybut jest tylko do odczytu.

Wyjście obciążenia klastra
Nazwa wyjścia obciążenia klastra. Ten atrybut jest tylko do odczytu.

Długość obciążenia klastra
Długość obciążenia klastra. Ten atrybut jest tylko do odczytu.

mru obciążenia klastra
Wartość ostatnio używanych kanałów obciążenia klastra. Ten atrybut jest tylko do odczytu.

Kolejka użycia obciążenia klastra
Obciążenie klastra używa wartości kolejki. Ten atrybut jest tylko do odczytu.

zdarzenie komendy
Określa, czy generowane są zdarzenia komend. Ten atrybut jest tylko do odczytu.

nazwa kolejki wejściowej komendy
Nazwa kolejki wejściowej komendy systemowej. Ten atrybut jest tylko do odczytu.

poziom komendy
Poziom komend obsługiwany przez menedżer kolejek. Ten atrybut jest tylko do odczytu.

Kontrola serwera komend
Określa, czy serwer komend powinien być uruchamiany automatycznie podczas uruchamiania menedżera kolejek. Ten atrybut jest tylko do odczytu.

Opcje połączenia
Opcje, które mają zastosowanie do metody połączenia. Wartością początkową jest MQCNO_NONE. W zależności od platformy mogą być dostępne następujące wartości dodatkowe:

- 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_q_mgr)
- MQCNO_RESTRICT_CONN_TAG_QSG

Identyfikator połączenia
Unikalny identyfikator, który umożliwia produktowi MQ niezawodną identyfikację aplikacji.

status połączenia
Wartość TRUE w przypadku połączenia z menedżerem kolejek. Ten atrybut jest tylko do odczytu.

Znacznik połączenia
Znacznik, który ma zostać powiązany z połączeniem. Ten atrybut można ustawić tylko wtedy, gdy nie jest nawiązane połączenie. Wartość początkowa jest pusta.

Sprzęt szyfrujący
Szczegóły konfiguracji sprzętu szyfrującego. Dla połączeń klienta MQI produktu MQ .

nazwa kolejki niedostarczonych komunikatów
Nazwa kolejki niedostarczonych komunikatów. Ten atrybut jest tylko do odczytu.

domyślna nazwa kolejki transmisji
Domyślna nazwa kolejki transmisji. Ten atrybut jest tylko do odczytu.

listy dystrybucyjne
Możliwość obsługi list dystrybucyjnych przez menedżera kolejek.

grupa dns
Nazwa grupy, do której powinien dołączyć program nasłuchujący TCP obsługujący transmisje przychodzące dla grupy współużytkowania kolejek, jeśli używana jest obsługa dynamicznych usług nazw domen (DNS) programu Workload Manager. Ten atrybut jest tylko do odczytu.

dns wlm
Określa, czy program nasłuchujący TCP, który obsługuje transmisje przychodzące dla grupy współużytkowania kolejek, powinien zostać zarejestrowany w programie Workload Manager for Dynamic Domain Name Services. Ten atrybut jest tylko do odczytu.

pierwszy rekord uwierzytelniania
Pierwszy z obiektów rekordu klasy ImqAuthentication(bez określonej kolejności), w którym połączenie rekordu ImqAuthenticationodwołuje się do tego obiektu. Dla połączeń klienta MQI produktu MQ .

pierwszy obiekt zarządzany
Pierwszy z obiektów klasy ImqObject(bez określonej kolejności), w którym połączenie ImqObject odwołuje się do tego obiektu. Wartością początkową jest zero.

Zdarzenie wstrzymania
Elementy sterujące-zablokuj zdarzenia. Ten atrybut jest tylko do odczytu.

Wersja adresu IP
Protokół IP (IPv4 lub IPv6), który ma być używany dla połączenia kanału. Ten atrybut jest tylko do odczytu.

repozytorium kluczy
Położenie pliku bazy danych kluczy, w którym przechowywane są klucze i certyfikaty. Dla połączeń IBM MQ MQI client .

liczba resetów klucza
Liczba niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji TLS przed ponownym negocjowaniem klucza tajnego. Ten atrybut dotyczy tylko połączeń klienckich używających wywołania MQCONNX. Patrz także ssl key reset count (licznik resetowania klucza ssl).

Zegar nasłuchiwania
Odstęp czasu (w sekundach) między próbami zrestartowania procesu nasłuchiwania podejmowanymi przez IBM MQ w przypadku awarii komunikacji APPC lub TCP/IP. Ten atrybut jest tylko do odczytu.

zdarzenie lokalne
Steruje lokalnymi zdarzeniami. Ten atrybut jest tylko do odczytu.

zdarzenie programu rejestrującego
Określa, czy generowane są zdarzenia dziennika odtwarzania. Ten atrybut jest tylko do odczytu.

Nazwa grupy LU
Ogólna nazwa jednostki logicznej, której powinien używać program nasłuchujący LU 6.2 obsługujący transmisje przychodzące dla grupy współużytkowania kolejek. Ten atrybut jest tylko do odczytu.

Nazwa LU
Nazwa jednostki logicznej, która ma być używana dla wychodzących transmisji LU 6.2 . Ten atrybut jest tylko do odczytu.

Przyrostek ramienia lu62
Przyrostek SYS1.PARMLIB podzbiór APPCPMxx, który nominuje LUADD dla tego inicjatora kanału. Ten atrybut jest tylko do odczytu.

Kanały lu62
Maksymalna liczba kanałów bieżących lub podłączonych klientów używających protokołu transmisji LU 6.2 . Ten atrybut jest tylko do odczytu.

maksymalna liczba aktywnych kanałów
Maksymalna liczba kanałów, które mogą być aktywne w dowolnym momencie. Ten atrybut jest tylko do odczytu.

Maksimum kanałów
Maksymalną liczba kanałów bieżących (w tym kanałów połączenia z serwerem z połączonymi klientami). Ten atrybut jest tylko do odczytu.

maksymalna liczba uchwytów
Maksymalna liczba uchwytów. Ten atrybut jest tylko do odczytu.

Maksymalna długość komunikatu
Maksymalna możliwa długość dowolnego komunikatu w dowolnej kolejce zarządzanej przez tego menedżera kolejek. Ten atrybut jest tylko do odczytu.

maksymalny priorytet
Maksymalny priorytet komunikatu. Ten atrybut jest tylko do odczytu.

Maks. liczba niezatw. kom.
Maksymalna liczba niezatwierdzonych komunikatów w jednostce lub pracy. Ten atrybut jest tylko do odczytu.

Rozliczanie MQI
Steruje kolekcjonowaniem informacji rozliczeniowych dla danych MQI. Ten atrybut jest tylko do odczytu.

Statystyka MQI
Steruje kolekcjonowaniem informacji monitorowania statystyk dla menedżera kolejek. Ten atrybut jest tylko do odczytu.

maksymalna liczba portów wychodzących
Wyższy koniec zakresu numerów portów, który ma być używany podczas wiązania kanałów wychodzących. Ten atrybut jest tylko do odczytu.

Minimalny port danych wychodzących
Dolny koniec zakresu numerów portów, który ma być używany podczas wiązania kanałów wychodzących. Ten atrybut jest tylko do odczytu.

Hasło
hasło powiązane z ID użytkownika

zdarzenie dotyczące wydajności
Steruje zdarzeniami wydajności. Ten atrybut jest tylko do odczytu.

rzeczy
Platforma, na której rezyduje menedżer kolejek. Ten atrybut jest tylko do odczytu.

Rozliczanie kolejek
Steruje kolekcjonowaniem informacji rozliczeniowych dla kolejek. Ten atrybut jest tylko do odczytu.

Monitorowanie kolejek
Steruje kolekcjonowaniem danych monitorowania bezpośredniego dla kolejek. Ten atrybut jest tylko do odczytu.

Statystyka kolejek
Steruje kolekcjonowaniem danych statystycznych dla kolejek. Ten atrybut jest tylko do odczytu.

Limit czasu odbierania
Przybliżony czas oczekiwania kanału komunikatów TCP/IP na odebranie danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego. Ten atrybut jest tylko do odczytu.

Minimalny limit czasu odbioru
Minimalny czas oczekiwania kanału TCP/IP na odbiór danych, w tym pulsów, od partnera przed powrotem do stanu nieaktywnego. Ten atrybut jest tylko do odczytu.

Typ limitu czasu odbierania
Kwalifikator stosowany do odbierania limitu czasu. Ten atrybut jest tylko do odczytu.

zdarzenie zdalne
Steruje zdarzeniami zdalnymi. Ten atrybut jest tylko do odczytu.

Nazwa repozytorium
Nazwa repozytorium. Ten atrybut jest tylko do odczytu.

Lista nazw repozytorium
Nazwa listy nazw repozytorium. Ten atrybut jest tylko do odczytu.

nazwa współużytkowanego menedżera kolejek
Określa, czy nazwy MQOPENs kolejki współużytkowanej, w której ObjectQMgrjest innym menedżerem kolejek w grupie współużytkowania kolejek, powinny być rozstrzygane jako otwarte kolejki współużytkowanej w lokalnym menedżerze kolejek. Ten atrybut jest tylko do odczytu.

Zdarzenie SSL
Określa, czy generowane są zdarzenia SSL. Ten atrybut jest tylko do odczytu.

Wymagane SSL FIPS
Określa, czy mają być używane tylko algorytmy z certyfikatem FIPS, jeśli szyfrowanie jest wykonywane w oprogramowaniu IBM MQ . Ten atrybut jest tylko do odczytu.

Licznik zerowania klucza SSL
Liczba niezaszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed ponownym negocjowaniem klucza tajnego. Ten atrybut jest tylko do odczytu.

Zdarzenie uruchomienia i zatrzymania
Steruje zdarzeniami uruchomienia i zatrzymania. Ten atrybut jest tylko do odczytu.

Interwał statystyki
Określa, jak często dane monitorowania statystyk są zapisywane w kolejce monitorowania. Ten atrybut jest tylko do odczytu.

Dostępność punktu synchronizacji
Dostępność uczestnictwa w punkcie synchronizacji. Ten atrybut jest tylko do odczytu.

Uwaga: Koordynowane przez menedżera kolejek globalne jednostki pracy nie są obsługiwane na platformie IBM i . 

Istnieje możliwość zaprogramowania jednostki pracy, koordynowanej zewnętrznie przez produkt IBM i, przy użyciu rodzimych wywołań systemowych _Rcommit i _Rback. Uruchom ten typ jednostki pracy, uruchamiając aplikację IBM MQ z kontrolą transakcji na poziomie zadania za pomocą komendy STRCMTCTL. Więcej informacji na ten temat zawiera sekcja Interfejsy do IBM i zewnętrznego menedżera punktów synchronizacji . Wycofywanie i zatwierdzanie są obsługiwane na platformie IBM i dla lokalnych jednostek pracy koordynowanych przez menedżer kolejek.

kanały tcp
Maksymalna liczba kanałów, które mogą być kanałami bieżącymi, lub klientów, które mogą być połączone, korzystających z protokołu transmisji TCP/IP. Ten atrybut jest tylko do odczytu.

Sprawdzanie połączenia TCP
Określa, czy narzędzie TCP KEEPALIVE ma być używane do sprawdzania, czy drugi koniec połączenia jest nadal dostępny. Ten atrybut jest tylko do odczytu.

Nazwa TCP
Nazwa jedynego lub domyślnego systemu TCP/IP, który ma być używany, w zależności od wartości typu stosu tcp. Ten atrybut jest tylko do odczytu.

Typ stosu TCP
Określa, czy inicjator kanału może używać tylko przestrzeni adresowej TCP/IP określonej w nazwie tcp, czy też może być powiązany z dowolnym wybranym adresem TCP/IP. Ten atrybut jest tylko do odczytu.

Zapis śledzenia trasy
Steruje rejestrowaniem informacji o śledzeniu trasy. Ten atrybut jest tylko do odczytu.

Interwał wyzwalacza
Odstęp czasu wyzwalacza. Ten atrybut jest tylko do odczytu.

ID użytkownika
Na platformach AIX and Linux : rzeczywisty identyfikator użytkownika aplikacji. Na platformach Windows : identyfikator użytkownika aplikacji.

Konstruktory

ImqQueueManager ();
Konstruktor domyślny.

ImqQueueManager (const ImqQueueManager & ***menedżer*** **);**
Konstruktor kopii. Status połączenia będzie miał wartość FALSE.

ImqQueueManager (const char * ***nazwa*** **);**
Ustawia nazwę ImqObject na *name*.

Destruktory

Po zniszczeniu obiektu menedżera ImqQueuejest on automatycznie odłączany.

Metody klasy (publiczne)

statyczne zachowanie MQLONG ();
Zwraca zachowanie.

void setBehavior(const MQLONG ***behavior*** **= 0);**
Ustawia zachowanie.

Metody obiektów (publiczne)

void operator = (const ImqQueueManager & ***mgr*** **);**
W razie potrzeby rozłącza się i kopiuje dane instancji z menedżera *mgr*. Status połączenia ma wartość FALSE.

ImqBoolean accountingConnOverride (MQLONG & statint);
Udostępnia kopię wartości nadpisania połączeń rozliczeniowych. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG accountingConnOverride ();
Zwraca wartość nadpisania połączeń rozliczeniowych bez wskazania możliwych błędów.

ImqBoolean accountingInterval (MQLONG & statint);
Udostępnia kopię wartości przedziału czasu rozliczania. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG accountingInterval ();
Zwraca wartość interwału rozliczania bez wskazania możliwych błędów.

ImqBoolean activityRecording (MQLONG & rec);
Udostępnia kopię wartości rejestrowania działania. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG activityRecording ();
Zwraca wartość zapisu aktywności bez wskazania możliwych błędów.

ImqBoolean adoptNewMCACheck (MQLONG & check);
Udostępnia kopię wartości sprawdzania adopcji nowego MCA. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG adoptNewMCACheck ();
Zwraca wartość sprawdzania adopcji nowego MCA bez wskazania możliwych błędów.

ImqBoolean adoptNewMCAType (MQLONG & typ);
Udostępnia kopię typu adopcji nowego MCA. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG adoptNewMCAType ();
Zwraca typ adopcji nowego MCA bez wskazania możliwych błędów.

QLONG authenticationType () konst;
Zwraca typ uwierzytelniania.

void setAuthenticationType (const MQLONG type = MQCSP_AUTH_NONE);
Ustawia typ uwierzytelniania.

ImqBoolean authorityEvent(MQLONG &*Zdarzenie*);
Udostępnia kopię stanu włączenia zdarzenia uprawnień. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG authorityEvent();
Zwraca stan włączenia zdarzenia uprawnienia bez wskazania możliwych błędów.

Funkcja ImqBoolean backout ();
Wycofuje niezatwierdzone zmiany. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean początek ();
Rozpoczyna jednostkę pracy. Opcje początku mają wpływ na zachowanie tej metody. Zwraca wartość TRUE, jeśli operacja powiodła się, ale zwraca również wartość TRUE, nawet jeśli bazowe wywołanie komendy MQBEGIN zwraca wartość MQRC_NO_EXTERNAL_UCZESTNICY lub MQRC_PARTICIPANT_NOT_AVAILABLE (oba powiązane z MQCC_WARNING).

MQLONG beginOptions() konst;
Zwraca opcje początku.

void setBeginOptions (const MQLONG *options* = MQBO_NONE);
Ustawia opcje początku.

ImqBoolean bridgeEvent (MQLONG & zdarzenie);
Udostępnia kopię wartości zdarzenia mostu. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG bridgeEvent ();
Zwraca wartość zdarzenia mostu bez wskazania możliwych błędów.

ImqBoolean channelAutoDefinition (MQLONG &*wartość*);
Udostępnia kopię wartości automatycznego definiowania kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelAuto-definicja ();
Zwraca wartość automatycznej definicji kanału bez wskazania możliwych błędów.

ImqBoolean channelAutoDefinitionEvent(MQLONG &*wartość*);
Udostępnia kopię wartości zdarzenia automatycznej definicji kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelAutoDefinitionEvent();
Zwraca wartość zdarzenia automatycznej definicji kanału bez wskazania możliwych błędów.

ImqBoolean channelAutoDefinitionExit(ImqString &*nazwa*);
Udostępnia kopię nazwy wyjścia automatycznej definicji kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString channelAutoDefinitionExit( );
Zwraca nazwę wyjścia automatycznej definicji kanału bez wskazania możliwych błędów.

ImqBoolean channelEvent (zdarzenie MQLONG &);
Udostępnia kopię wartości zdarzenia kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelEvent();
Zwraca wartość zdarzenia kanału bez wskazania możliwych błędów.

MQLONG channelInitiatorAdapters ();
Zwraca wartość adapterów inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorAdapters (adaptery MQLONG &);
Udostępnia kopię wartości adapterów inicjatora kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelInitiatorControl ();
Zwraca wartość uruchamiania inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorControl (MQLONG & init);
Udostępnia kopię wartości startowej sterowania inicjatorem kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelInitiatorDispatchers ();
Zwraca wartość przekaźników inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorProgramy rozsyłające (MQLONG & programy rozsyłające);
Udostępnia kopię wartości programów rozsyłających inicjatora kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelInitiatorTraceAutoStart ();
Zwraca wartość automatycznego uruchamiania śledzenia inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorTraceAutoStart (MQLONG & auto);
Udostępnia kopię wartości automatycznego uruchamiania śledzenia inicjatora kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelInitiatorTraceTableSize ();
Zwraca wartość wielkości tabeli śledzenia inicjatora kanału bez wskazania możliwych błędów.

ImqBoolean channelInitiatorTraceTableWielkość (MQLONG & wielkość);
Udostępnia kopię wartości wielkości tabeli śledzenia inicjatora kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean channelMonitoring (MQLONG & monchl);
Udostępnia kopię wartości monitorowania kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelMonitoring ();
Zwraca wartość monitorowania kanału bez wskazania możliwych błędów.

ImqBoolean channelReference(ImqChannel * & ***pchannel*** **);**
Udostępnia kopię odwołania do kanału. Jeśli odwołanie do kanału jest niepoprawne, ustawia wartość *pchannel* na null. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqChannel * channelReference( );
Zwraca odwołanie do kanału bez wskazania możliwych błędów.

ImqBoolean setChannelReference (ImqChannel & ***kanał*** **);**
Ustawia odwołanie do kanału. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setChannelReference (ImqChannel * ***kanał*** **= 0);**
Ustawia lub resetuje odwołanie do kanału. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean channelStatistics (MQLONG & statchl);
Udostępnia kopię wartości statystyki kanału. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG channelStatistics ();
Zwraca wartość statystyki kanału bez wskazania możliwych błędów.

ImqBoolean characterSet(MQLONG & ***ccsid*** **);**
Udostępnia kopię zestawu znaków. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG characterSet();
Zwraca kopię zestawu znaków bez wskazania możliwych błędów.

Liczba operacji MQLONG clientSslKeyReset() konst;
Zwraca wartość licznika resetowania klucza SSL używaną w połączeniach klienckich.

void setClientSslKeyResetCount(const MQLONG count);
Ustawia licznik resetowania klucza SSL używanego w połączeniach klienckich.

ImqBoolean clusterSenderMonitoring (MQLONG & monacls);
Udostępnia kopię wartości domyślnej monitorowania nadawcy klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterSenderMonitoring ();
Zwraca wartość domyślną monitorowania nadawcy klastra bez wskazania możliwych błędów.

ImqBoolean clusterSenderStatistics (MQLONG & statacls);
Udostępnia kopię wartości statystyki nadawcy klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterSenderStatystyki ();
Zwraca wartość statystyki nadajnika klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkloadData (ImqString & *dane*);
Udostępnia kopię danych wyjścia obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString clusterWorkloadData ();
Zwraca dane wyjścia obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkloadExit (ImqString & *nazwa*);
Udostępnia kopię nazwy wyjścia obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString clusterWorkloadExit ();
Zwraca nazwę wyjścia obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkloadLength (MQLONG & *długość*);
Udostępnia kopię długości obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterWorkloadLength ();
Zwraca długość obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkLoadMRU (MQLONG & mru);
Udostępnia kopię wartości ostatnio używanych kanałów obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterWorkLoadMRU ();
Zwraca wartość ostatnio używanych kanałów obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean clusterWorkLoadUseQ (MQLONG & useq);
Udostępnia kopię wartości kolejki użycia obciążenia klastra. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG clusterWorkLoadUseQ ();
Zwraca wartość kolejki użycia obciążenia klastra bez wskazania możliwych błędów.

ImqBoolean commandEvent (MQLONG & zdarzenie);
Udostępnia kopię wartości zdarzenia komendy. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG commandEvent ();
Zwraca wartość zdarzenia komendy bez wskazania możliwych błędów.

ImqBoolean commandInputQueueName(ImqString & *nazwa*);
Udostępnia kopię nazwy kolejki wejściowej komend. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString commandInputQueueName( );
Zwraca nazwę kolejki wejściowej komendy bez wskazania możliwych błędów.

ImqBoolean commandLevel(MQLONG & *poziom*);
Udostępnia kopię poziomu komendy. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG commandLevel();
Zwraca poziom komendy bez wskazania możliwych błędów.

MQLONG commandServerControl ();
Zwraca wartość startową serwera komend bez wskazania możliwych błędów.

ImqBoolean commandServerControl (MQLONG & server);
Udostępnia kopię wartości startowej sterowania serwerem komend. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean commit ();
Zatwierdza niezatwierdzone zmiany. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean connect ();
Nawiązuje połączenie z menedżerem kolejek o podanej nazwie ImqObject , domyślnie jest to lokalny menedżer kolejek. Aby nawiązać połączenie z konkretnym menedżerem kolejek, przed nawiązaniem połączenia należy użyć metody ImqObject setName . Jeśli istnieje odwołanie do kanału, jest ono używane do przekazywania informacji o definicji kanału do MQCONNX w MQCD. Parametr ChannelType w tabeli MQCD jest ustawiony na wartość MQCHT_CLNTCONN. informacje o odwołaniach do kanału, które są istotne tylko dla połączeń klienckich, są ignorowane w przypadku połączeń

z serwerem. Opcje połączenia mają wpływ na zachowanie tej metody. Ta metoda ustawia status połączenia na TRUE w przypadku powodzenia. Zwraca nowy status połączenia.

Jeśli istnieje pierwszy rekord uwierzytelniania, łańcuch rekordów uwierzytelniania jest używany do uwierzytelniania certyfikatów cyfrowych dla bezpiecznych kanałów klienta.

Z tym samym menedżerem kolejek można połączyć więcej niż jeden obiekt menedżera ImqQueue. Wszystkie używają tego samego uchwytu połączenia MQHCONN i funkcji współużytkowania jednostki pracy dla połączenia powiązanego z wątkiem. Pierwszy menedżer ImqQueue, który nawiązał połączenie, uzyskuje uchwyt MQHCONN. Ostatni rozłączony menedżer ImqQueuewykonuje operację MQDISC.

W przypadku programu wielowątkowego zaleca się użycie oddzielnego obiektu menedżera ImqQueuedla każdego wątku.

ImqBinary connectionId () konst;
Zwraca identyfikator połączenia.

ImqBinary connectionTag () konst;
Zwraca znacznik połączenia.

ImqBoolean setConnectionsetConnection (const MQBYTE128 *znacznik* = 0);
Ustawia znacznik połączenia. Jeśli *znacznik* ma wartość zero, usuwa znacznik połączenia. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setConnectionZnacznik (const ImqBinary & *znacznik*);
Ustawia znacznik połączenia. Długość danych znacznika *znacznik* musi wynosić zero (w celu wyczyszczenia znacznika połączenia). lub MQ_CONN_TAG_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

MQLONG connectOptions() konst;
Zwraca opcje połączenia.

void setConnectOptions (const MQLONG *options* = MQCNO_NONE);
Ustawia opcje połączenia.

ImqBoolean connectionStatus() konst;
Zwraca status połączenia.

ImqString cryptographicHardware ( );
Zwraca sprzęt szyfrujący.

ImqBoolean setCryptographicHardware (const char * *sprzęt* = 0);
Ustawia sprzęt szyfrujący. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean deadLetterQueueName(ImqString & *nazwa*);
Udostępnia kopię nazwy kolejki niedostarczonych komunikatów. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString deadLetterQueueName( );
Zwraca kopię nazwy kolejki niedostarczonych komunikatów bez wskazania możliwych błędów.

ImqBoolean defaultTransmissionQueueName(ImqString & *nazwa*);
Udostępnia kopię domyślnej nazwy kolejki transmisji. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString defaultTransmissionQueueName( );
Zwraca domyślną nazwę kolejki transmisji bez wskazania możliwych błędów.

Funkcja ImqBoolean disconnect ();
Rozłącza się z menedżerem kolejek i ustawia status połączenia na FALSE. Zamyka wszystkie obiekty ImqProcess i ImqQueue powiązane z tym obiektem i odłącza ich odwołania do połączenia przed rozłączeniem. Jeśli więcej niż jeden obiekt menedżera ImqQueuejest połączony z tym samym menedżerem kolejek, tylko ostatni rozłączony obiekt wykonuje fizyczne rozłączenie; inni wykonują logiczne rozłączenie. Niezatwierdzone zmiany są zatwierdzane tylko przy fizycznym rozłączeniu.

Ta metoda zwraca wartość TRUE w przypadku powodzenia. Jeśli jest wywoływana, gdy nie ma istniejącego połączenia, kod powrotu ma również wartość true.

ImqBoolean distributionLists(MQLONG & ***obsługa*** **);**
Udostępnia kopię wartości list dystrybucyjnych. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG distributionLists();
Zwraca wartość listy dystrybucyjnej bez wskazania możliwych błędów.

ImqBoolean dnsGroup (ImqString & grupa);
Udostępnia kopię nazwy grupy DNS. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString dnsGroup ( );
Zwraca nazwę grupy DNS bez wskazania możliwych błędów.

ImqBoolean dnsWlm (MQLONG & wlm);
Udostępnia kopię wartości WLM DNS. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG dnsWlm ();
Zwraca wartość DNS WLM bez wskazania możliwych błędów.

Rekord ImqAuthentication* firstAuthenticationRecord () konst;
Zwraca pierwszy rekord uwierzytelniania.

void setFirstAuthenticationRecord (const ImqAuthenticationRecord * ***air*** **= 0);**
Ustawia pierwszy rekord uwierzytelniania.

ImqObject * firstManagedObject () konst;
Zwraca pierwszy obiekt zarządzany.

ImqBoolean inhibitEvent(MQLONG & ***zdarzenie*** **);**
Udostępnia kopię stanu włączenia zdarzenia blokady. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG inhibitEvent();
Zwraca stan włączenia zdarzenia zablokowanego bez wskazania możliwych błędów.

ImqBoolean ipAddressWersja (MQLONG & wersja);
Udostępnia kopię wartości wersji adresu IP. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG ipAddressVersion ();
Zwraca wartość wersji adresu IP bez wskazania możliwych błędów.

ImqBoolean keepAlive (MQLONG & keepalive);
Udostępnia kopię wartości sprawdzania połączenia. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG keepAlive ();
Zwraca wartość sprawdzania połączenia bez wskazania możliwych błędów.

ImqString keyRepository ( );
Zwraca repozytorium kluczy.

ImqBoolean setKeyRepozytorium (znak konst * ***repozytorium*** **= 0);**
Ustawia repozytorium kluczy. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean listenerTimer (MQLONG & timer);
Udostępnia kopię wartości licznika czasu nasłuchiwania. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG listenerTimer ();
Zwraca wartość licznika czasu nasłuchiwania bez wskazania możliwych błędów.

ImqBoolean localEvent(MQLONG & ***Zdarzenie*** **);**
Udostępnia kopię stanu włączenia zdarzenia lokalnego. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG localEvent();
Zwraca stan włączenia zdarzenia lokalnego bez wskazania możliwych błędów.

ImqBoolean loggerEvent (MQLONG & count);
Udostępnia kopię wartości zdarzenia programu rejestrującego. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG loggerEvent ();
Zwraca wartość zdarzenia programu rejestrującego bez wskazania możliwych błędów.

ImqBoolean luGroupNazwa (ImqString & nazwa);.
Udostępnia kopię nazwy grupy jednostek logicznych. Zwraca TRUE w przypadku powodzenia

ImqString luGroupName ();
Zwraca nazwę grupy jednostek logicznych bez wskazania możliwych błędów.

ImqBoolean lu62ARMSuffix (ImqString & przyrostek);
Udostępnia kopię przyrostka ARM LU62 . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString lu62ARMSuffix ( );
Zwraca przyrostek ARM LU62 bez wskazania możliwych błędów.

ImqBoolean luName (ImqString & nazwa);
Udostępnia kopię nazwy jednostki logicznej. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString luName ( );
Zwraca nazwę jednostki logicznej bez wskazania możliwych błędów.

ImqBoolean maximumActivekanałów (MQLONG & channels);
Udostępnia kopię wartości maksymalnej liczby aktywnych kanałów. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumActiveKanały ();
Zwraca wartość maksymalnej liczby aktywnych kanałów bez wskazania możliwych błędów.

ImqBoolean maximumCurrentkanałów (MQLONG & channels);
Udostępnia kopię wartości maksymalnej liczby bieżących kanałów. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumCurrentkanałów ();
Zwraca wartość maksymalnej liczby bieżących kanałów bez wskazania możliwych błędów.

ImqBoolean maximumHandles(MQLONG & *liczba*);
Udostępnia kopię maksymalnej liczby uchwytów. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumHandles();
Zwraca maksymalną liczbę uchwytów bez wskazania możliwych błędów.

ImqBoolean maximumLu62Channels (MQLONG & channels);
Kopia maksymalnej liczby kanałów LU62 . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumLu62Channels ();.
Zwraca maksymalną liczbę kanałów LU62 bez wskazania możliwych błędów.

ImqBoolean maximumMessageDługość (MQLONG & *długość*);
Udostępnia kopię maksymalnej długości komunikatu. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumMessageLength ();
Zwraca maksymalną długość komunikatu bez wskazania możliwych błędów.

ImqBoolean maximumPriority(MQLONG & *priorytet*);
Udostępnia kopię maksymalnego priorytetu. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumPriority();
Zwraca kopię priorytetu maksymalnego bez wskazania możliwych błędów.

ImqBoolean maximumTcpKanały (MQLONG & channels);
Udostępnia kopię wartości maksymalnej liczby kanałów TCP. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumTcpChannels ();
Zwraca wartość maksymalnej liczby kanałów TCP bez wskazania możliwych błędów.

ImqBoolean maximumUncommittedkomunikatów (MQLONG & *liczba*);
Udostępnia kopię maksymalnej liczby niezatwierdzonych komunikatów. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG maximumUncommittedkomunikatów ();
Zwraca maksymalną liczbę niezatwierdzonych komunikatów bez wskazania możliwych błędów.

ImqBoolean mqiAccounting (MQLONG & statint);
Udostępnia kopię wartości rozliczania MQI. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG mqiAccounting ();
Zwraca wartość rozliczania MQI bez wskazania możliwych błędów.

ImqBoolean mqiStatistics (MQLONG & statmqi);
Udostępnia kopię wartości statystyki MQI. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG mqiStatistics ();
Zwraca wartość statystyki MQI bez wskazania możliwych błędów.

ImqBoolean outboundPortMax (MQLONG & max);
Udostępnia kopię maksymalnej wartości portu wychodzącego. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG outboundPortMax ();
Zwraca maksymalną wartość portu wychodzącego bez wskazania możliwych błędów.

ImqBoolean outboundPortMin (MQLONG & min);
Udostępnia kopię minimalnej wartości portu wychodzącego. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG outboundPortMin ();
Zwraca minimalną wartość portu wychodzącego bez wskazania możliwych błędów.

ImqBinary password () konst;
Zwraca hasło używane w połączeniach klienckich.

ImqBoolean setPassword (const ImqString & hasło);
Ustawia hasło używane w połączeniach klienckich.

ImqBoolean setPassword (const char * = 0 hasło);
Ustawia hasło używane w połączeniach klienckich.

ImqBoolean setPassword (const ImqBinary & password);
Ustawia hasło używane w połączeniach klienckich.

ImqBoolean performanceEvent(MQLONG & *zdarzenie*);
Udostępnia kopię stanu włączenia zdarzenia wydajności. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG performanceEvent();
Zwraca stan włączenia zdarzenia wydajności bez wskazania możliwych błędów.

ImqBoolean platform (MQLONG & *platforma*);
Udostępnia kopię platformy. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

platforma MQLONG ();
Zwraca platformę bez wskazania możliwych błędów.

ImqBoolean queueAccounting (MQLONG & acctq);
Udostępnia kopię wartości rozliczania kolejki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueAccounting ();
Zwraca wartość rozliczania kolejki bez wskazania możliwych błędów.

ImqBoolean queueMonitoring (MQLONG & monq);
Udostępnia kopię wartości monitorowania kolejki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueMonitoring ();
Zwraca wartość monitorowania kolejki bez wskazania możliwych błędów.

ImqBoolean queueStatistics (MQLONG & statq);
Udostępnia kopię wartości statystyki kolejki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG queueStatistics ();
Zwraca wartość statystyki kolejki bez wskazania możliwych błędów.

ImqBoolean receiveTimeout (MQLONG & limit_czasu);
Udostępnia kopię wartości limitu czasu odbierania. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG receiveTimeout ();
Zwraca wartość limitu czasu odbioru bez wskazania możliwych błędów.

ImqBoolean receiveTimeoutMin (MQLONG & min);
Udostępnia kopię minimalnej wartości limitu czasu odbierania. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG receiveTimeoutMin ();
Zwraca minimalną wartość limitu czasu odbioru bez wskazania możliwych błędów.

ImqBoolean receiveTimeoutTyp (MQLONG & typ);
Udostępnia kopię typu limitu czasu odbierania. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG receiveTimeoutType ();
Zwraca typ limitu czasu odbioru bez wskazania możliwych błędów.

ImqBoolean remoteEvent(MQLONG & *Zdarzenie*);
Udostępnia kopię stanu włączenia zdarzenia zdalnego. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG remoteEvent();
Zwraca stan włączenia zdarzenia zdalnego bez wskazania możliwych błędów.

ImqBoolean repositoryName(ImqString & *nazwa*);
Udostępnia kopię nazwy repozytorium. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString repositoryName( );
Zwraca nazwę repozytorium bez wskazania możliwych błędów.

ImqBoolean repositoryNamelistName (ImqString & *nazwa*);
Udostępnia kopię nazwy listy nazw repozytorium. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString repositoryNamelistName ();
Zwraca kopię nazwy listy nazw repozytorium bez wskazania możliwych błędów.

ImqBoolean sharedQueueQueueManagerNazwa (MQLONG & nazwa);
Udostępnia kopię wartości nazwy współużytkowanego menedżera kolejek. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG sharedQueueQueueManagerName ();
Zwraca wartość nazwy menedżera kolejek współużytkowanych bez wskazania możliwych błędów.

ImqBoolean sslEvent (MQLONG & zdarzenie);
Udostępnia kopię wartości zdarzenia SSL. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG sslEvent ();
Zwraca wartość zdarzenia SSL bez wskazania możliwych błędów.

ImqBoolean sslFips (MQLONG & sslfips);
Udostępnia kopię wartości SSL FIPS. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG sslFips ();
Zwraca wartość SSL FIPS bez wskazania możliwych błędów.

ImqBoolean sslKeyResetCount (MQLONG & licznik);
Udostępnia kopię wartości licznika resetowania klucza SSL. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG sslKeyResetCount ();
Zwraca wartość licznika resetowania klucza SSL bez wskazania możliwych błędów.

ImqBoolean startStopZdarzenie (MQLONG & *Zdarzenie*);
Udostępnia kopię stanu włączenia zdarzenia start-stop. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG startStopEvent ();
Zwraca stan włączenia zdarzenia start-stop bez wskazania możliwych błędów.

ImqBoolean statisticsInterval (MQLONG & statint);
Udostępnia kopię wartości przedziału czasu statystyki. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG statisticsInterval ();
Zwraca wartość interwału statystycznego bez wskazania możliwych błędów.

ImqBoolean syncPointDostępność (MQLONG & ***sync*** **);**
Udostępnia kopię wartości dostępności punktu synchronizacji. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG syncPointAvailability ();
Zwraca kopię wartości dostępności punktu synchronizacji bez wskazania możliwych błędów.

ImqBoolean tcpName (ImqString & nazwa);
Udostępnia kopię nazwy systemu TCP. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqString tcpName ( );
Zwraca nazwę systemu TCP bez wskazania możliwych błędów.

ImqBoolean tcpStack-typ (MQLONG & type);
Udostępnia kopię typu stosu TCP. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG tcpStackTyp ();
Zwraca typ stosu TCP bez wskazania możliwych błędów.

ImqBoolean traceRouteRecording (MQLONG & routerec);
Udostępnia kopię wartości zapisu trasy śledzenia. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG traceRouteRecording ();
Zwraca wartość zapisu trasy śledzenia bez wskazania możliwych błędów.

ImqBoolean triggerInterval(MQLONG & ***interwał*** **);**
Udostępnia kopię odstępu czasu wyzwalacza. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

MQLONG triggerInterval();
Zwraca odstęp czasu wyzwalacza bez wskazania możliwych błędów.

ImqBinary userId () konst;
Zwraca identyfikator użytkownika używany w połączeniach klienckich.

ImqBoolean setUserId (const ImqString & id);
Ustawia identyfikator użytkownika używany w połączeniach klienckich.

ImqBoolean setUserIdentyfikator (const char * = 0 id);
Ustawia identyfikator użytkownika używany w połączeniach klienckich.

ImqBoolean setUserId (const ImqBinary & id);
Ustawia identyfikator użytkownika używany w połączeniach klienckich.

Metody obiektów (zabezpieczone)

void setFirstManagedObject (const ImqObject * ***object*** **= 0);**
Ustawia pierwszy obiekt zarządzany.

Dane obiektu (zabezpieczone)

MQHCONN ***omconn***
Uchwyt połączenia IBM MQ (ma znaczenie tylko wtedy, gdy status połączenia ma wartość TRUE).

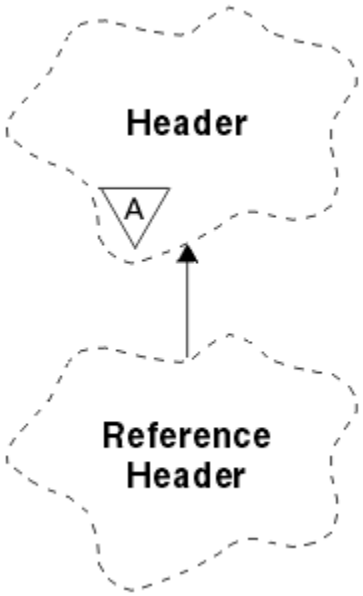
Kody przyczyny

- MQRC_ATTRIBUTE_LOCKED (atrybut mqrc_locked)
- BŁĄD MQRC_ENVIRONMENT_ERROR
- MQRC_FUNCTION_NOT_SUPPORTED (nieobsługiwana funkcja MQRAL)
- BŁĄD MQRC_REFERENCE_ERROR

- (kody przyczyny dla MQBACK)
- (kody przyczyny dla komendy MQBEGIN)
- (kody przyczyny dla MQCMIT)
- (kody przyczyny dla MQCONNX)
- (kody przyczyny dla MQDISC)
- (kody przyczyny dla MQCONN)

Klasa C++ nagłówka ImqReference

Ta klasa obudowuje funkcje struktury danych MQRMH.



Rysunek 34. Klasa nagłówka ImqReference

Ta klasa odnosi się do wywołań MQI wymienionych w sekcji “ImqReferenceOdwołanie do nagłówka” na stronie 1856.

- “Atrybuty obiektu” na stronie 1940
- “Konstruktory” na stronie 1941
- “Przeciążone metody ImqItem” na stronie 1941
- “Metody obiektów (publiczne)” na stronie 1941
- “Dane obiektu (zabezpieczone)” na stronie 1942
- “Kody przyczyny” na stronie 1942

Atrybuty obiektu

środowisko docelowe
Środowisko dla miejsca docelowego. Wartością początkową jest łańcuch pusty.

Nazwa miejsca docelowego
Nazwa miejsca docelowego danych. Wartością początkową jest łańcuch pusty.

Identyfikator instancji
Identyfikator instancji. Wartość binarna (MQBYTE24) o długości MQ_OBJECT_INSTANCE_ID_LENGTH. Wartością początkową jest MQOII_NONE.

długość logiczna
Logiczna lub zamierzona długość danych komunikatu następujących po tym nagłówku. Wartością początkową jest zero.

przesunięcie logiczne
Przesunięcie logiczne dla następujących danych komunikatu, które mają być interpretowane w kontekście danych jako całość, w miejscu docelowym. Wartością początkową jest zero.

przesunięcie logiczne 2
Rozszerzenie wysokiego rzędu do przesunięcia logicznego. Wartością początkową jest zero.

Typ odwołania
Typ odwołania. Wartością początkową jest łańcuch pusty.

Środowisko źródłowe
Środowisko dla źródła. Wartością początkową jest łańcuch pusty.

NAZWA ŹRÓDŁA
Nazwa źródła danych. Wartością początkową jest łańcuch pusty.

Konstruktory

ImqReferenceNagłówek ();
Konstruktor domyślny.

ImqReferenceHeader (const ImqReferenceHeader & *header*);
Konstruktor kopii.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & *komunikat*);
Wstawia strukturę danych MQRMH do buforu komunikatów na początku, przenosząc istniejące dane komunikatu dalej i ustawia format *msg* na wartość MQFMT_REF_MSG_HEADER.

Więcej informacji na ten temat zawiera opis metody klasy ImqHeader w sekcji "Klasa C++ ImqHeader" na stronie 1885 .

virtual ImqBoolean pasteIn (ImqMessage & *komunikat*);
Odczytuje strukturę danych MQRMH z buforu komunikatów.

Aby operacja zakończyła się pomyślnie, komunikat ImqMessage musi mieć format MQFMT_REF_MSG_HEADER.

Więcej informacji na ten temat zawiera opis metody klasy ImqHeader w sekcji "Klasa C++ ImqHeader" na stronie 1885 .

Metody obiektów (publiczne)

void operator = (const ImqReferenceNagłówek & *nagłówek*);
Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

ImqString destinationEnvironment () konst;
Zwraca kopię środowiska docelowego.

void setDestinationEnvironment (const char * *environment* = 0);
Ustawia środowisko docelowe.

ImqString destinationName () konst;
Zwraca kopię nazwy miejsca docelowego.

void setDestinationNazwa (const char * *nazwa* = 0);
Ustawia nazwę miejsca docelowego.

ImqBinary instanceId () konst;
Zwraca kopię identyfikatora instancji.

ImqBoolean setInstanceId (const ImqBinary & *id*);
Ustawia identyfikator instancji. Długość danych *token* musi wynosić 0 lub MQ_OBJECT_INSTANCE_ID_LENGTH. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

void setInstanceId (const MQBYTE24 *id* = 0);
Ustawia identyfikator instancji. *id* może mieć wartość zero, co jest równoważne określeniu wartości MQOII_NONE. Jeśli *id* ma wartość różną od zera, musi adresować bajty danych binarnych o wartości MQ_OBJECT_INSTANCE_ID_LENGTH. Jeśli używane są predefiniowane wartości, takie jak MQOII_NONE, konieczne może być wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQOII_NONE.

MQLONG logicalLength () konst;
Zwraca długość logiczną.

void setLogicalLength (const MQLONG *długość*);
Ustawia długość logiczną.

MQLONG logicalOffset () konst;
Zwraca przesunięcie logiczne.

void setLogicalOffset (const MQLONG *offset*);
Ustawia przesunięcie logiczne.

MQLONG logicalOffset2 () konst;
Zwraca przesunięcie logiczne 2.

void setLogicalOffset2 (const MQLONG *przesunięcie*);
Ustawia przesunięcie logiczne 2.

ImqString referenceType () konst;
Zwraca kopię typu odwołania.

void setReferenceType (const char * *nazwa* = 0);
Ustawia typ odwołania.

ImqString sourceEnvironment () konst;
Zwraca kopię środowiska źródłowego.

void setSourceEnvironment (const char * *environment* = 0);
Ustawia środowisko źródłowe.

ImqString sourceName () konst;
Zwraca kopię nazwy źródła.

void setSourceName (const char * *nazwa* = 0);
Ustawia nazwę źródła.

Dane obiektu (zabezpieczone)

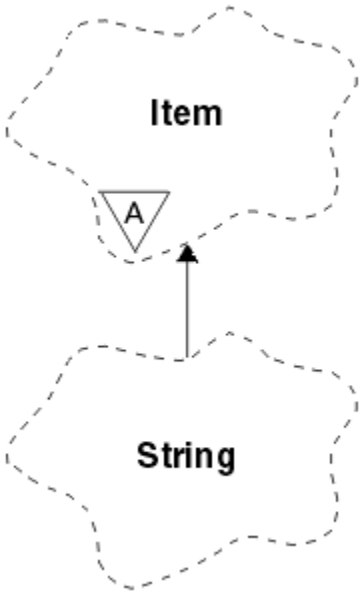
MQRMH *omqrmh*
Struktura danych MQRMH.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR
- MQRC_STRUC_LENGTH_ERROR
- BŁĄD MQRC_STRUC_ID_ERROR
- MQRC_INSUFFICIENT_DATA
- MQRC_INCONSISTENT_FORMAT,
- BŁĄD MQRC_ENCODING_ERROR

Klasa ImqString C++

Ta klasa umożliwia przechowywanie łańcuchów znakowych i manipulowanie nimi w przypadku łańcuchów zakończonych znakiem o kodzie zero.



Rysunek 35. Klasa ImqString

Łańcucha ImqString należy używać zamiast znaku **char *** w większości sytuacji, gdy parametr wywołuje funkcję **char ***.

- "Atrybuty obiektu" na stronie 1943
- "Konstruktory" na stronie 1943
- "Metody klasy (publiczne)" na stronie 1944
- "Przeciążone metody ImqItem" na stronie 1944
- "Metody obiektów (publiczne)" na stronie 1944
- "Metody obiektów (zabezpieczone)" na stronie 1947
- "Kody przyczyny" na stronie 1947

Atrybuty obiektu

zn.
Znaki w **pamięci masowej** , które poprzedzają końcowe wartości null.

długość
Liczba bajtów w **znakach**. Jeśli nie ma **pamięci masowej**, **długość** wynosi zero. Wartością początkową jest zero.

przechowywanie
Ulotna tablica bajtów o dowolnej wielkości. W **pamięci masowej** po **znakach**zawsze musi występować końcowa wartość NULL, aby można było wykryć koniec **znaków** . Metody zapewniają zachowanie tej sytuacji, ale podczas bezpośredniego ustawiania bajtów w tablicy należy upewnić się, że po modyfikacji istnieje końcowa wartość pusta. Początkowo nie ma atrybutu **storage** .

Konstruktory

ImqString( );
Konstruktor domyślny.

ImqString(const ImqString & *łańcuch*);
Konstruktor kopii.

ImqString(znak konst *c*);
Znaki składają się z *c*.

ImqString(const char * *tekst*);
Znaki są kopiowane z *tekstu*.

ImqString(const void * *buffer*, const size_t *długość*);
Kopiuje *długość* bajtów począwszy od *buforu* i przypisuje je do **znaków**. Podstawienie jest wykonywane dla każdego skopiowanego znaku o kodzie zero. Znakiem podstawianym jest kropka (.). Nie są brane pod uwagę żadne inne niedrukowalne ani niewyświetlane znaki, które zostały skopiowane.

Metody klasy (publiczne)

static ImqBoolean copy (char * *bufor-docelowy*, const size_t *długość*, const char * *bufor-źródłowy*, const char *pad* = 0);
Kopiuje do *długości* bajtów z *buforu źródłowego* do *buforu docelowego*. Jeśli liczba znaków w *buforze źródłowym* jest niewystarczająca, zapełnia pozostałą przestrzeń w *buforze docelowym* znakami *dopełniającymi* . Wartość *source-buffer* może wynosić zero. Parametr *destination-buffer* może mieć wartość zero, jeśli parametr *length* ma również wartość zero. Wszystkie kody błędów zostały utracone. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

static ImqBoolean copy (char * *bufor-docelowy*, const size_t *długość*, const char * *source-buffer*, ImqError & *error-object*, const char *pad* = 0);
Kopiuje do *długości* bajtów z *buforu źródłowego* do *buforu docelowego*. Jeśli liczba znaków w *buforze źródłowym* jest niewystarczająca, zapełnia pozostałą przestrzeń w *buforze docelowym* znakami *dopełniającymi* . Wartość *source-buffer* może wynosić zero. Parametr *destination-buffer* może mieć wartość zero, jeśli parametr *length* ma również wartość zero. Wszystkie kody błędów są ustawiane w elemencie *error-object*. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & *komunikat*);
Kopiuje **znaki** do buforu komunikatów, zastępując istniejącą treść. Ustawia *msg* **format** na wartość MQFMT_STRING.

Więcej informacji na ten temat zawiera opis metody klasy nadrzędnej.

wirtualny ImqBoolean pasteIn (ImqMessage & *komunikat*);
Ustawia **znaki** , przesyłając pozostałe dane z buforu komunikatów, zastępując istniejące **znaki**.

Aby operacja zakończyła się pomyślnie, **kodowanie** obiektu *msg* musi mieć wartość MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT do MQENC_NATIVE.

Aby operacja zakończyła się pomyślnie, **format** komunikatu ImqMessage musi mieć wartość MQFMT_STRING.

Więcej informacji na ten temat zawiera opis metody klasy nadrzędnej.

Metody obiektów (publiczne)

char & operator [] (const size_t *przesunięcie*) konst;
Odnosi się do znaku o przesunięciu *offset* w **pamięci**. Upewnij się, że odpowiedni bajt istnieje i jest adresowalny.

Operator ImqString () (const size_t *przesunięcie*, const size_t *długość* = 1) konst;
Zwraca podłańcuch, kopiując bajty z **znaków** , zaczynając od *przesunięcia*. Jeśli *długość* wynosi zero, zwraca resztę **znaków**. Jeśli kombinacja *przesunięcia* i *długości* nie powoduje utworzenia odwołania w obrębie **znaków**, zwraca pusty łańcuch ImqString.

void operator = (const ImqString & *łańcuch*);
Kopiuje dane instancji z *łańcucha*, zastępując istniejące dane instancji.

ImqString operator + (const char *c*) konst;
Zwraca wynik dopisania *c* do **znaków**.

ImqString operator + (const char * *tekst*) konst;
Zwraca wynik dopisania *tekstu* do **znaków**. Można to również odwrócić. Na przykład:

```
strOne + "string two" ;
"string one" + strTwo ;
```

Uwaga: Chociaż większość kompilatorów akceptuje łańcuch **strOne + "string two";** Microsoft Visual C++ wymaga łańcucha **strOne + (char *) "string two";**

ImqString operator + (const ImqString & *string1*) konst;
Zwraca wynik dodawania łańcucha *string1* do **znaków**.

ImqString operator + (const double *liczba*) konst;
Zwraca wynik dopisania *liczby* do **znaków** po konwersji na tekst.

ImqString operator + (const long *liczba*) konst;
Zwraca wynik dopisania *liczby* do **znaków** po konwersji na tekst.

void operator + = (const znak *c*);
Dodaje łańcuch *c* do **znaków**.

void operator + = (const char * *tekst*);
Dopisuje *tekst* do **znaków**.

void operator + = (const ImqString & *łańcuch*);
Dodaje *łańcuch* do **znaków**.

void operator + = (const double *liczba*);
Po konwersji na tekst dopisuje *liczbę* do **znaków** .

void operator + = (const long *liczba*);
Po konwersji na tekst dopisuje *liczbę* do **znaków** .

znak operatora * () konst;
Zwraca adres pierwszego bajtu w **pamięci masowej**. Ta wartość może być zerem i jest ulotna. Tej metody należy używać tylko w trybie tylko do odczytu.

ImqBoolean operator < (const ImqString & *łańcuch*) konst;
Porównuje **znaki** z *łańcuchami* przy użyciu metody **compare** . Wynik ma wartość TRUE, jeśli jest mniejszy, lub FALSE, jeśli jest większy lub równy.

Operator ImqBoolean > (const ImqString & *łańcuch*) konst;
Porównuje **znaki** z *łańcuchami* przy użyciu metody **compare** . Wynik ma wartość TRUE, jeśli jest większy, lub FALSE, jeśli jest mniejszy lub równy.

ImqBoolean operator < = (const ImqString & *łańcuch*) konst;
Porównuje **znaki** z *łańcuchami* przy użyciu metody **compare** . Wynikiem jest TRUE, jeśli jest mniejsze lub równe, lub FALSE, jeśli większe niż.

ImqBoolean operator > = (const ImqString & *łańcuch*) konst;
Porównuje **znaki** z *łańcuchami* przy użyciu metody **compare** . Wynikiem jest wartość TRUE, jeśli jest większa lub równa, lub wartość FALSE, jeśli jest mniejsza niż.

ImqBoolean operator == (const ImqString & *łańcuch*) konst;
Porównuje **znaki** z *łańcuchami* przy użyciu metody **compare** . Zwraca TRUE lub FALSE.

ImqBoolean operator! = (const ImqString & *łańcuch*) konst;
Porównuje **znaki** z *łańcuchami* przy użyciu metody **compare** . Zwraca TRUE lub FALSE.

short compare (const ImqString & *string*) konst;
Porównuje **znaki** z *łańcuchami*. Wynikiem jest zero, jeśli **znaki** są równe, ujemne, jeśli mniejsze niż i dodatnie, jeśli większe niż. W porównaniu rozróżniana jest wielkość liter. Łańcuch ImqString o wartości NULL jest traktowany jako mniejszy niż łańcuch ImqStringo wartości innej niż NULL.

ImqBoolean copyOut(char * *buffer*, const size_t *długość*, const char *pad* = 0);
Kopiuje do *długości* bajtów z **znaków** do *buforu*. Jeśli liczba **znaków** jest niewystarczająca, zapełnia pozostałą przestrzeń w *buforze* znakami *dopełniającymi* . Parametr *buffer* może mieć wartość zero, jeśli parametr *length* ma również wartość zero. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

size_t copyOut(long &*liczba*) konst;

Ustawia *liczbę* z **znaków** po konwersji z tekstu i zwraca liczbę znaków biorących udział w konwersji. Wartość zero oznacza, że konwersja nie została przeprowadzona, a wartość *liczba* nie jest ustawiona. Przekształcalna sekwencja znaków musi zaczynać się od następujących wartości:

```
<blank(s)>
<+|->
digit(s)
```

size_t copyOut(ImqString &*token*, const char *c* = '') konst;

Jeśli **znaki** zawierają jeden lub więcej znaków innych niż *c*, identyfikuje leksem jako pierwszą ciągłą sekwencję takich znaków. W tym przypadku *token* jest ustawiony na tę sekwencję, a zwrócona wartość jest sumą liczby znaków wiodących *c* i liczby bajtów w sekwencji. W przeciwnym razie zwraca zero i nie ustawia *token*.

size_t cutOut(long &*liczba*);

Ustawia wartość *number* (liczba) dla metody **copy** , ale usuwa również z pola **characters** liczbę bajtów wskazanych przez wartość zwracaną. Na przykład łańcuch przedstawiony w poniższym przykładzie można podzielić na trzy liczby za pomocą funkcji **cutOut** (*liczba*). trzy razy:

```
strNumbers = "-1 0   +55 "

while ( strNumbers.cutOut( number ) );
number becomes -1, then 0, then 55
leaving strNumbers == " "
```

size_t cutOut(ImqString &*token*, const char *c* = ' '

Ustawia *Token* jako metodę **copyOut** i usuwa **znaki** znaki *strToken* oraz wszystkie znaki *C* , które poprzedzają znaki *Token* . Jeśli *c* nie jest spacją, usuwa znaki *c* , które bezpośrednio występują po znakach *token* . Zwraca liczbę usuniętych znaków. Na przykład łańcuch przedstawiony w poniższym przykładzie można podzielić na trzy leksemy za pomocą funkcji **cutOut** (*token*). trzy razy:

```
strText = " Program Version 1.1 "

while ( strText.cutOut( token ) );

// token becomes "Program", then "Version",
// then "1.1" leaving strText == " "
```

Poniższy przykład przedstawia sposób analizowania nazwy ścieżki 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 &*łańcuch*);

Wyszukuje dokładne dopasowanie dla *łańcucha* w dowolnym miejscu w obrębie **znaków**. Jeśli nie zostanie znaleziony żaden wynik, zwraca FALSE. W przeciwnym razie zwraca TRUE. Jeśli *łańcuch* ma wartość null, zwraca TRUE.

ImqBoolean find (const ImqString &*łańcuch*, size_t &*przesunięcie*);

Wyszukuje dokładne dopasowanie dla *łańcucha* w obrębie **znaków** , począwszy od przesunięcia *przesunięcia* . Jeśli *łańcuch* ma wartość null, zwraca TRUE bez aktualizowania *przesunięcia*. Jeśli dopasowanie nie zostanie znalezione, zwracana jest wartość FAŁSZ (wartość *przesunięcie* mogła zostać zwiększona). Jeśli zostanie znalezione dopasowanie, zwraca TRUE i aktualizuje *przesunięcie* do przesunięcia *łańcucha* w obrębie **znaków**.

Długość size_t () konst;
Zwraca **długość**.

ImqBoolean pasteIn(const double *liczba*, const char * *format* = "%f");
Po konwersji na tekst dopisuje *liczbę* do **znaków** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Specyfikacja *format* jest używana do formatowania konwersji zmiennopozycyjnej. Jeśli jest określony, musi być odpowiedni do użycia z **printf** i liczbami zmiennopozycyjną, na przykład **%.3f**.

ImqBoolean pasteIn(const long *liczba*);
Po konwersji na tekst dopisuje *liczbę* do **znaków** . Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

ImqBoolean pasteIn(const void * *buffer*, const size_t *długość*);
Dołącza *długość* bajtów z *buforu* do **znaków**i dodaje końcowe wartości null. Zastępuje wszystkie skopiowane znaki o kodzie zero. Znakiem podstawianym jest kropka (.). Nie są brane pod uwagę żadne inne niedrukowalne lub niewyświetlane znaki, które zostały skopiowane. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean set (const char * *bufor*, const size_t *długość*);
Ustawia **znaki** z pola znakowego o stałej długości, które może zawierać wartość null. W razie potrzeby dodaje wartość null do znaków z pola o stałej długości. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqBoolean setStorage(const size_t *długość*);
Przydziela (lub ponownie przydziela) **pamięć masową**. Zachowuje wszystkie oryginalne **znaki**, w tym wszystkie końcowe znaki null, jeśli nadal jest na nie miejsce, ale nie inicjuje dodatkowej pamięci.

Ta metoda zwraca wartość TRUE w przypadku powodzenia.

Pamięć masowa size_t () konst;
Zwraca liczbę bajtów w **pamięci masowej**.

size_t stripLeading(const char *c* = '');
Usuwa znaki wiodące *c* z **znaków** i zwraca usuniętę liczbę.

size_t stripTrailing(const char *c* = '');
Usuwa znaki końcowe *c* z **znaków** i zwraca liczbę usuniętych.

ImqString upperCase() konst;
Zwraca kopię **znaków**zapisaną wielkimi literami.

Metody obiektów (zabezpieczone)

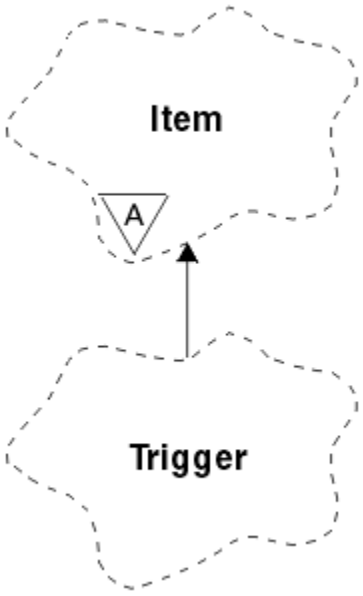
ImqBoolean przypisać (const ImqString & *łańcuch*);
Odpowiednik równoważnej metody **operator =** , ale niewirtualnej. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

Kody przyczyny

- MQRC_DATA_OBCIĘTE
- MQRC_NULL_POINTER (wskaźnik wartości NULL)
- MQRC_STORAGE_NIEDOSTĘPNY
- BŁĄD MQRC_BUFFER_ERROR
- MQRC_INCONSISTENT_FORMAT,

Klasa ImqTrigger C++

Ta klasa hermetyzuje strukturę danych MQTM (komunikat wyzwalacza).



Rysunek 36. Klasa ImqTrigger

Obiekty tej klasy są zwykle używane przez program monitora wyzwalacza. Zadaniem programu monitora wyzwalacza jest oczekiwanie na te konkretne komunikaty i wykonywanie na nich działań w celu upewnienia się, że inne aplikacje IBM MQ są uruchamiane, gdy oczekują na nie komunikaty.

Przykład użycia można znaleźć w przykładowym programie IMQSTRG.

- "Atrybuty obiektu" na stronie 1948
- "Konstruktory" na stronie 1949
- "Przeciążone metody ImqItem" na stronie 1949
- "Metody obiektów (publiczne)" na stronie 1949
- "Dane obiektu (zabezpieczone)" na stronie 1950
- "Kody przyczyny" na stronie 1950

Atrybuty obiektu

Identyfikator aplikacji
Tożsamość aplikacji, która wysłała komunikat. Wartością początkową jest łańcuch pusty.

Typ aplikacji
Typ aplikacji, która wysłała komunikat. Wartością początkową jest zero. Możliwe są następujące wartości dodatkowe:

- 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
- Tabela MQAT_USER_LAST

Dane środowiska
Dane środowiska dla procesu. Wartością początkową jest łańcuch pusty.

Nazwa procesu
Nazwa procesu. Wartością początkową jest łańcuch pusty.

Nazwa kolejki
Nazwa kolejki, która ma zostać uruchomiona. Wartością początkową jest łańcuch pusty.

Dane wyzwalacza
Dane wyzwalacza dla procesu. Wartością początkową jest łańcuch pusty.

Dane użytkownika
Dane użytkownika dla procesu. Wartością początkową jest łańcuch pusty.

Konstruktory

ImqTrigger( );
Konstruktor domyślny.

ImqTrigger(const ImqTrigger & *wyzwalacz*);
Konstruktor kopii.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut (ImqMessage & *komunikat*);
Zapisuje strukturę danych MQTM w buforze komunikatów, zastępując istniejącą treść. Ustawia format *msg* na wartość MQFMT_TRIGGER.

Więcej informacji na ten temat zawiera opis metody klasy ImqItem w sekcji “Klasa ImqItem C++” na stronie 1889 .

virtual ImqBoolean pasteIn (ImqMessage & *komunikat*);
Odczytuje strukturę danych MQTM z buforu komunikatów.

Aby operacja zakończyła się pomyślnie, komunikat ImqMessage musi mieć format MQFMT_TRIGGER.

Więcej informacji na ten temat zawiera opis metody klasy ImqItem w sekcji “Klasa ImqItem C++” na stronie 1889 .

Metody obiektów (publiczne)

void operator = (const ImqTrigger & *wyzwalacz*);
Kopiuje dane instancji z *wyzwalacza*, zastępując istniejące dane instancji.

ImqString applicationId () konst;
Zwraca kopię identyfikatora aplikacji.

void setApplicationId (const char * *id*);
Ustawia identyfikator aplikacji.

MQLONG applicationType () konst;
Zwraca typ aplikacji.

void setApplicationType (const MQLONG *typ*);
Ustawia typ aplikacji.

ImqBoolean copyOut (MQTMC2 * *ptmc2*);
Hermetyzuje strukturę danych MQTM, która jest strukturą odbieraną w kolejkach inicjujących. Wypełnia równoważną strukturę danych MQTMC2 udostępnioną przez program wywołujący i ustawia pole QMgrName (które nie jest obecne w strukturze danych MQTM) na wartość pustą. Struktura

danych MQTMC2 jest tradycyjnie używana jako parametr dla aplikacji uruchamianych przez monitor wyzwalacza. Ta metoda zwraca wartość TRUE w przypadku powodzenia.

ImqString environmentData () konst;
Zwraca kopię danych środowiska.

void setEnvironmentData (const char * *data*);
Ustawia dane środowiska.

ImqString processName () konst;
Zwraca kopię nazwy procesu.

void setProcessName (const char * *nazwa*);
Ustawia nazwę procesu, dopełnianą odstępami, na 48 znaków.

ImqString queueName () konst;
Zwraca kopię nazwy kolejki.

void setQueueName (const char * *nazwa*);
Ustawia nazwę kolejki, dopełnianie odstępami na 48 znaków.

ImqString triggerData () konst;
Zwraca kopię danych wyzwalacza.

void setTriggerData (const char * *dane*);
Ustawia dane wyzwalacza.

ImqString userData () konst;
Zwraca kopię danych użytkownika.

void setUserData (const char * *data*);
Ustawia dane użytkownika.

Dane obiektu (zabezpieczone)

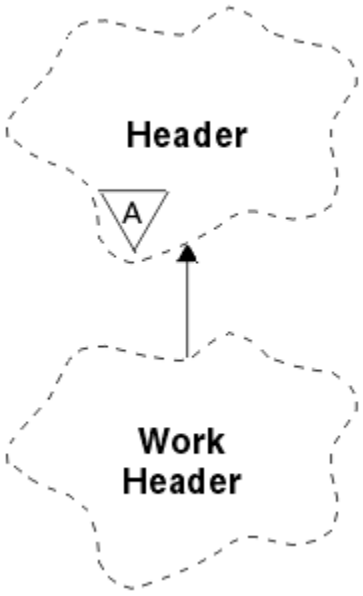
MQTM *omqtm*
Struktura danych MQTM.

Kody przyczyny

- MQRC_NULL_POINTER (wskaźnik wartości NULL)
- MQRC_INCONSISTENT_FORMAT,
- BŁĄD MQRC_ENCODING_ERROR
- BŁĄD MQRC_STRUC_ID_ERROR

Klasa C++ nagłówka ImqWork

Ta klasa hermetyzuje konkretne funkcje struktury danych MQWIH.



Rysunek 37. Klasa nagłówka ImqWork

Obiekty tej klasy są używane przez aplikacje umieszczają komunikaty w kolejce zarządzanej przez program z/OS Workload Manager.

- "Atrybuty obiektu" na stronie 1951
- "Konstruktory" na stronie 1951
- "Przeciążone metody ImqItem" na stronie 1951
- "Metody obiektów (publiczne)" na stronie 1952
- "Dane obiektu (zabezpieczone)" na stronie 1952
- "Kody przyczyny" na stronie 1952

Atrybuty obiektu

znacznik komunikatu
Znacznik komunikatu dla menedżera obciążenia z/OS o długości MQ_MSG_TOKEN_LENGTH. Wartością początkową jest MQMTOK_NONE.

Nazwa usługi
32-znakowa nazwa procesu. Nazwa jest początkowo pusta.

krok usługi
8-znakowa nazwa kroku w procesie. Nazwa jest początkowo pusta.

Konstruktory

ImqWorkHeader ();
Konstruktor domyślny.

ImqWorkHeader (const ImqWorkHeader & *header*);
Konstruktor kopii.

Przeciążone metody ImqItem

virtual ImqBoolean copyOut(ImqMessage & *komunikat*);
Wstawia strukturę danych MQWIH na początku buforu komunikatów, przenosząc istniejące dane komunikatu dalej, a następnie ustawia parametr *msg* **format** na wartość MQFMT_WORK_INFO_HEADER.

Więcej szczegółów zawiera opis metody klasy macierzystej.

virtual ImqBoolean pasteIn(ImqMessage & *komunikat*);
Odczytuje strukturę danych MQWIH z buforu komunikatów.

Aby operacja zakończyła się pomyślnie, obiekt *msg* musi mieć kodowanie MQENC_NATIVE. Pobieranie komunikatów z opcją MQGMO_CONVERT do MQENC_NATIVE.

Parametr ImqMessage musi mieć format MQFMT_WORK_INFO_HEADER.

Więcej szczegółów zawiera opis metody klasy macierzystej.

Metody obiektów (publiczne)

void operator = (const ImqWorkNagłówek & *nagłówek*);
Kopiuje dane instancji z *nagłówka*, zastępując istniejące dane instancji.

ImqBinary messageToken () konst;
Zwraca **znacznik komunikatu**.

ImqBoolean setMessageToken (const ImqBinary & *token*);
Ustawia **znacznik komunikatu**. Długość danych elementu *token* musi mieć wartość zero lub wartość MQ_MSG_TOKEN_LENGTH. Zwraca TRUE, jeśli operacja zakończyła się pomyślnie.

void setMessageToken (const MQBYTE16 *token* = 0);
Ustawia **znacznik komunikatu**. Parametr *token* może mieć wartość zero, co odpowiada określeniu wartości MQMTOK_NONE. Jeśli *token* ma wartość niezerową, musi adresować bajty danych binarnych (MQ_MSG_TOKEN_LENGTH).

Jeśli używane są wartości predefiniowane, takie jak MQMTOK_NONE, może być konieczne wykonanie rzutowania w celu zapewnienia zgodności sygnatury, na przykład (MQBYTE *) MQMTOK_NONE.

ImqString serviceName () konst;
Zwraca **nazwę usługi**wraz ze spacjami kończącymi.

void setServiceName (const char * *nazwa*);
Ustawia **nazwę usługi**.

ImqString serviceStep () konst;
Zwraca **krok usługi**, w tym końcowe odstępy.

void setServiceStep (const znak * *krok*);
Ustawia **krok usługi**.

Dane obiektu (zabezpieczone)

MQWIH-omqwih
Struktura danych MQWIH.

Kody przyczyny

- MQRC_BINARY_DATA_LENGTH_ERROR

Właściwości obiektów IBM MQ classes for JMS

Wszystkie obiekty w IBM MQ classes for JMS mają właściwości. Różne właściwości mają zastosowanie do różnych typów obiektów. Różne właściwości mają różne dozwolone wartości, a symboliczne wartości właściwości różnią się w zależności od narzędzia administracyjnego i kodu programu.

IBM MQ classes for JMS udostępnia narzędzia do ustawiania i odpytywania właściwości obiektów za pomocą narzędzia administracyjnego IBM MQ JMS , Eksploratora IBM MQ lub aplikacji. Wiele właściwości dotyczy tylko konkretnego podzbioru typów obiektów.

Informacje na temat korzystania z narzędzia administracyjnego IBM MQ JMS zawiera sekcja Konfigurowanie obiektów JMS za pomocą narzędzia administracyjnego.

Tabela 868 na stronie 1953 zawiera krótki opis każdej właściwości i przedstawia dla każdej właściwości typy obiektów, do których ma ona zastosowanie. Typy obiektów są identyfikowane za pomocą słów

kluczowych. Informacje na temat tych obiektów zawiera sekcja Konfigurowanie obiektów JMS za pomocą narzędzia administracyjnego .

Liczby odnoszą się do uwag na końcu tabeli. Patrz także “Zależności między właściwościami obiektów IBM MQ classes for JMS” na stronie 1956.

Właściwość składa się z pary nazwa-wartość w formacie:

```
PROPERTY_NAME(property_value)
```

Tematy w tej sekcji zawierają listę, dla każdej właściwości, nazwę właściwości i krótki opis, a także poprawne wartości właściwości używane w narzędziu administracyjnym. i metodę set, która jest używana do ustawiania wartości właściwości w aplikacji. W tematach przedstawiono również poprawne wartości właściwości dla każdej właściwości oraz odwzorowanie między symbolicznymi wartościami właściwości używanymi w narzędziu i ich programowalnymi odpowiednikami.

W nazwach właściwości nie jest rozróżniana wielkość liter i są one ograniczone do zestawu rozpoznawanych nazw wyświetlanych w tych tematach.

Tabela 868. Nazwy właściwości i odpowiednie typy obiektów

Właściwość	**Postać krótka**	**Typ obiektu**							
		CF	**QCF,**	**TCF,**	**Q**	**T**	**XACF**	**XAQCF ,**	**XATCF,**
“APPLICATIONNAME” na stronie 1958	APPNAME	Y	Y	Y			Y	Y	Y
“WYJĄTEK ASYNCEXCEPTION” na stronie 1959	AEX	Y	Y	Y			Y	Y	Y
“BALOPCJE” na stronie 1960	OPCJE	Y	Y	Y			Y	Y	Y
“TYP BALTYPE” na stronie 1960	TYPE	Y	Y	Y			Y	Y	Y
“LIMIT_CZASU” na stronie 1961	TIMEOUT	Y	Y	Y			Y	Y	Y
“BROKERCCDURSUBQ” na stronie 1961 [1]	CCDSUB					Y			
“BROKERCCSUBQ” na stronie 1962 [1]	CCSUB	Y		Y			Y		Y
“BROKERCONQ” na stronie 1962 [1]	BCON	Y		Y			Y		Y
“BROKERDURSUBQ” na stronie 1963 [1]	BDSUB					Y			
“BROKERPUBQ” na stronie 1963 [1]	BPUB	Y		Y		Y	Y		Y
“BROKERPUBQMGR” na stronie 1964 [1]	BPQM					Y			
“BROKERQMGR” na stronie 1964 [1]	BQM	Y		Y			Y		Y
“BROKERSUBQ” na stronie 1964 [1]	BSUB	Y		Y			Y		Y
“BROKERVER” na stronie 1965 [1]	BVER	T [2]		T [2]		Y	Y		Y
“CCDTURL” na stronie 1966 [3]	CCDT	Y	Y	Y			Y	Y	Y
“CCSID” na stronie 1966	CCS	Y	Y	Y	Y	Y	Y	Y	Y
“CHANNEL” na stronie 1967 [3]	CHAN	Y	Y	Y			Y	Y	Y
“CLEANUP” na stronie 1967 [1]	CL	Y		Y			Y		Y

Tabela 868. Nazwy właściwości i odpowiednie typy obiektów (kontynuacja)

Właściwość	**Postać krótka**	**Typ obiektu**							
		CF	**QCF,**	**TCF,**	**Q**	**T**	**XACF**	**XAQCF ,**	**XATCF,**
"CLEANUPINT" na stronie 1968 [1]	CLINT	Y		Y			Y		Y
"LISTA NAZW POŁĄCZEŃ" na stronie 1968	LISTA CNLISTA	Y	Y	Y					
"CLIENTRECONNECTOPTIONS" na stronie 1968	CROPT	Y	Y	Y					
"CLIENTRECONNECTTIMEOUT" na stronie 1969	CRT	Y	Y	Y					
"CLIENTID" na stronie 1970	CID	T [2]	Y	T [2]			Y	Y	Y
"CLONESUPP" na stronie 1970	CLS	Y		Y			Y		Y
"COMPHDR" na stronie 1971	HC	Y		Y			Y		Y
"COMPMSG" na stronie 1971	MC	Y	Y	Y			Y	Y	Y
"CONNOPT" na stronie 1972	CNOPT	Y	Y	Y			Y	Y	Y
"CONNTAG" na stronie 1973	CNTAG	Y	Y	Y			Y	Y	Y
"OPIS" na stronie 1973	DESC	T [2]	Y	T [2]	Y	Y	Y	Y	Y
"DIRECTAUTH" na stronie 1974	DAUTH	T [2]		T [2]					
"ENCODING" na stronie 1974	ENC				Y	Y			
"EXPIRY" na stronie 1975	EXP				Y	Y			
"FAILIFQUIESCE" na stronie 1975	FIQ	Y	Y	Y	Y	Y	Y	Y	Y
"HOSTNAME" na stronie 1976	HOST	T [2]	Y	T [2]			Y	Y	Y
"LOCALADDRESS" na stronie 1977	LA	T [2]	Y	T [2]			Y	Y	Y
"STYL NAZWY MAPY" na stronie 1977	MNST	Y	Y	Y			Y	Y	Y
"MAXBUFFSIZE" na stronie 1978	MBSZ	T [2]		T [2]					
"MDREAD" na stronie 1978	MDR				Y	Y			
"MDWRITE" na stronie 1979	MDW				Y	Y			
"MDMSGCTX" na stronie 1979	MDCTX				Y	Y			
"MSGBATCHSZ" na stronie 1980 [1]	MBS	Y	Y	Y			Y	Y	Y
"MSGBODY" na stronie 1981	MBODY				Y	Y			
"MSGRETENTION" na stronie 1981	MRET	Y	Y				Y	Y	
"MSGSELECTION" na stronie 1982 [1]	MSEL	Y		Y			Y		Y
"MULTICAST" na stronie 1982	MCAST	T [2]		T [2]		Y			
"OPTIMISTICPUBLICATION" na stronie 1983 [1]	OPTPUB	Y		Y					
"OUTCOMENOTIFICATION" na stronie 1984 [1]	NOTIFY	Y		Y					

Tabela 868. Nazwy właściwości i odpowiednie typy obiektów (kontynuacja)

Właściwość	**Postać krótka**	**Typ obiektu**							
		CF	**QCF,**	**TCF,**	**Q**	**T**	**XACF**	**XAQCF ,**	**XATCF,**
"PERSISTENCE" na stronie 1984	PER				Y	Y			
"POLLINGINT" na stronie 1985 [1]	PINT	Y	Y	Y			Y	Y	Y
"PORT" na stronie 1985	PORT	T [2]	Y	T [2]			Y	Y	Y
"PRIORYTET" na stronie 1986	PRI				Y	Y			
"PROCESSDURATION" na stronie 1986 [1]	PROCDUR	Y		Y					
"PROVIDERVERSION" na stronie 1987	PVER	Y	Y	Y			Y	Y	Y
"PROXYHOSTNAME" na stronie 1989	PHOST	T [2]		T [2]					
"PROXYPORT" na stronie 1990	PPORT	T [2]		T [2]					
"PUBACKINT" na stronie 1990 [1]	PAI	Y		Y			Y		Y
"PUTASYNCALLOWED" na stronie 1991	PAALD				Y	Y			
"QMANAGER" na stronie 1991	QMGR	Y	Y	Y	Y		Y	Y	Y
"QUEUE" na stronie 1992	QU				Y				
"READAHEADALLOWED" na stronie 1992	RAALD				Y	Y			
"READAHEADCLOSEPOLICY" na stronie 1993	RACP				Y	Y			
"RECEIVECCSID" na stronie 1993	RCCS				Y	Y			
"RECEIVECONVERSION" na stronie 1994	RCNV				Y	Y			
"RECEIVEISOLATION" na stronie 1994 [1]	RCVISOL	Y		Y					
"RECEXIT" na stronie 1995	RCX	Y	Y	Y			Y	Y	Y
"RECEXITINIT" na stronie 1995	RCXI	Y	Y	Y			Y	Y	Y
"REPLYTOSTYLE" na stronie 1996	RTOST				Y	Y			
"RESCANINT" na stronie 1996 [1]	RINT	Y	Y				Y	Y	
"SECEXIT" na stronie 1997	SCX	Y	Y	Y			Y	Y	Y
"SECEXITINIT" na stronie 1998	SCXI	Y	Y	Y			Y	Y	Y
"SENDCHECKCOUNT" na stronie 1998	SCC	Y	Y	Y			Y	Y	Y
"SENDEXIT" na stronie 1998	SDX	Y	Y	Y			Y	Y	Y
"SENDEXITINIT" na stronie 1999	SDXI	Y	Y	Y			Y	Y	Y
"SHARECONVALLOWED" na stronie 1999	SCALD	Y	Y	Y			Y	Y	Y

Tabela 868. Nazwy właściwości i odpowiednie typy obiektów (kontynuacja)

Właściwość	Postać krótka	Typ obiektu							
		CF	QCF,	TCF,	Q	T	XACF	XAQCF,	XATCF,
"SPARSESUBS" na stronie 2000 [1]	SSUBS	Y		Y					
"SSLCIPHERSUITE" na stronie 2001	SCPHS	Y	Y	Y			Y	Y	Y
"SSLCRL" na stronie 2001	SCRL	Y	Y	Y			Y	Y	Y
"SSLFIPSREQUIRED" na stronie 2002	SFIPS	Y	Y	Y			Y	Y	Y
"SSLPEERNAME" na stronie 2002	SPEER	Y	Y	Y			Y	Y	Y
"SSLRESETCOUNT" na stronie 2003	SRC	Y	Y	Y			Y	Y	Y
"STATREFRESHINT" na stronie 2003 [1]	SRI	Y		Y			Y		Y
"SUBSTORE" na stronie 2004 [1]	SS	Y		Y			Y		Y
"SYNCPOINTALLGETS" na stronie 2004	SPAG	Y	Y	Y			Y	Y	Y
"TARGCLIENT" na stronie 2005	TC				Y	Y			
"TARGCLIENTMATCHING" na stronie 2005	TCM	Y	Y				Y	Y	
"TEMPMODEL" na stronie 2006	TM	Y	Y				Y	Y	
"TEMPQPREFIX" na stronie 2006	TQP	Y	Y				Y	Y	
"TEMPTOPICPREFIX" na stronie 2007	TTP	Y		Y			Y		Y
"TOPIC" na stronie 2007	TOP					Y			
"TRANSPORT" na stronie 2008	TRAN	T [2]	Y	T [2]			Y	Y	Y
"WILDCARDFORMAT" na stronie 2008	WCFMT	Y		Y			Y		Y

Uwaga:

1. Ta właściwość może być używana z wersją 70 produktu IBM MQ classes for JMS , ale nie ma wpływu na aplikację połączoną z menedżerem kolejek produktu IBM WebSphere MQ 7.0 , chyba że właściwość PROVIDERVERSION fabryki połączeń jest ustawiona na numer wersji mniejszy niż 7.
2. W przypadku korzystania z połączenia w czasie rzeczywistym z brokerem obsługiwane są tylko właściwości BROKERVER, CLIENTID, DESCRIPTION, DIRECTAUTH, HOSTNAME, LOCALADDRESS, MAXBUFFSIZE, MULTICAST, PORT, PROXYHOSTNAME, PROXYPORT i TRANSPORT.
3. Właściwości CCDTURL i CHANNEL obiektu nie mogą być ustawione jednocześnie.

Zależności między właściwościami obiektów IBM MQ classes for JMS

Poprawność niektórych właściwości zależy od konkretnych wartości innych właściwości.

Ta zależność może występować w następujących grupach właściwości:

- Właściwości klienta
- Właściwości połączenia z brokerem w czasie rzeczywistym
- Łańcuchy inicjowania wyjścia

Właściwości klienta

W przypadku połączenia z menedżerem kolejek następujące właściwości mają zastosowanie tylko wtedy, gdy właściwość TRANSPORT ma wartość CLIENT:

- HOSTNAME
- PORT
- CHANNEL
- LOCALADDRESS
- CCDTURL
- CCSID
- COMPHDR
- COMPMSG
- RECEXIT
- RECEXITINIT
- SECEXIT
- SECEXITINIT
- SENDEXIT
- SENDEXITINIT
- SHARECONVALLOWED
- SSLCIPHERSUITE
- SSLCRL
- SSLFIPSREQUIRED
- SSLPEERNAME
- SSLRESETCOUNT
- APPLICATIONNAME

Nie można ustawić wartości dla tych właściwości za pomocą narzędzia administracyjnego, jeśli TRANSPORT ma wartość BIND.

Jeśli właściwość TRANSPORT ma wartość CLIENT, wartością domyślną właściwości BROKERVER jest V1 , a wartością domyślną właściwości PORT jest 1414. Jeśli wartość parametru BROKERVER lub PORT zostanie ustawiona jawnie, późniejsza zmiana wartości parametru TRANSPORT nie spowoduje przesłonięcia wybranych opcji.

Właściwości połączenia z brokerem w czasie rzeczywistym

Tylko następujące właściwości mają znaczenie, jeśli TRANSPORT ma wartość DIRECT lub DIRECTHTTP:

- BROKERVER
- CLIENTID
- OPIS
- DIRECTAUTH
- HOSTNAME
- LOCALADDRESS
- MAXBUFFSIZE
- MULTICAST (obsługiwane tylko dla DIRECT)
- PORT
- PROXYHOSTNAME (obsługiwane tylko dla DIRECT)
- PROXYPORT (obsługiwane tylko dla DIRECT)

Jeśli TRANSPORT ma wartość DIRECT lub DIRECTHTTP, wartością domyślną właściwości BROKERVER jest V2, a wartością domyślną właściwości PORT jest 1506. Jeśli wartość parametru BROKERVER lub PORT zostanie ustawiona jawnie, późniejsza zmiana wartości parametru TRANSPORT nie spowoduje przesłonięcia wybranych opcji.

Łańcuchy inicjowania wyjścia

Nie należy ustawiać żadnego łańcucha inicjowania wyjścia bez podawania odpowiedniej nazwy wyjścia. Właściwości inicjowania wyjścia są następujące:

- RECEXITINIT
- SECEXITINIT
- SENDEXITINIT

Na przykład podanie wartości `RECEXITINIT(myString)` bez określenia wartości `RECEXIT(some.exit.classname)` spowoduje wystąpienie błędu.

Odsyłacze pokrewne

“TRANSPORT” na stronie 2008
Rodzaj połączenia z menedżerem kolejek lub brokerem.

APPLICATIONNAME

Aplikacja może ustawić nazwę identyfikującą jej połączenie z menedżerem kolejek. Ta nazwa aplikacji jest wyświetlana w komendzie **DISPLAY CONN MQSC/PCF** (gdzie pole jest nazywane **APPLTAG**). lub na ekranie **Połączenia aplikacji** w Eksploratorze IBM MQ (gdzie pole ma nazwę **App name**).

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : APPLICATIONNAME

Skrócona nazwa narzędzia administracyjnego JMS : APPNAME

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setAppNazwa ()
- MQConnectionFactory.getAppNazwa ()

Wartości

Dowolny poprawny łańcuch, który nie jest dłuższy niż 28 znaków. Dłuższe nazwy są dopasowywane do dopasowania przez usunięcie początkowych nazw pakietów, jeśli jest to konieczne. Jeśli na przykład klasą wywołującą jest `com.example.MainApp`, używana jest pełna nazwa, ale jeśli klasą wywołującą jest `com.example.dictionaryAndThesaurus.multilingual.mainApp`, używana jest nazwa `multilingual.mainApp` , ponieważ jest to najdłuższa kombinacja nazwy klasy i nazwy pakietu po prawej stronie, która pasuje do dostępnej długości.

Jeśli sama nazwa klasy ma więcej niż 28 znaków, zostanie obcięta w celu dopasowania.
Na przykład łańcuch `com.example.mainApplicationForSecondTestCase` będzie mieć postać `mainApplicationForSecondTest`.

z/OS W systemie z/OS, wartość APPNAME w:

- Tryb powiązań jest ignorowany, jeśli jest ustawiony, a jeśli jest ustawiony, może być tylko pusty.
- Tryb klienta może być ustawiony i używany.

WYJĄTEK ASYNCEXCEPTION

Ta właściwość określa, czy obiekt IBM MQ classes for JMS informuje obiekt ExceptionListener tylko w przypadku zerwania połączenia, czy też w przypadku wystąpienia dowolnego wyjątku asynchronicznie względem wywołania interfejsu API języka JMS . Dotyczy to wszystkich połączeń utworzonych na podstawie tej ConnectionFactory , dla których zarejestrowano obiekt nasłuchiwania wyjątków (ExceptionListener).

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : ASYNCEXCEPTION

Skrócona nazwa narzędzia administracyjnego JMS : AEX

Dostęp programowy

Procedury ustawiające/pobierające

- MQConnectionFactory.setAsyncWyjątki ()
- MQConnectionFactory.getAsync-wyjątki ()

Wartości

ASYNC_EXCEPTIONS_ALL

Wszystkie wyjątki wykryte asynchronicznie, poza zasięgiem synchronicznego wywołania interfejsu API, oraz wszystkie wyjątki zerwania połączenia są wysyłane do obiektu ExceptionListener.

Tabela 869. Wszystkie wyjątki asynchroniczne: środowiska i pokrewne nazwy stałych

Środowisko	Wartość
JMS narzędzie administracyjne	ALL
Zautomatyzowane	WMQCONSTANTS.ASYNC_EXCEPTIONS_ALL = -1
IBM MQ Explorer	Wszystkie

ASYNC_EXCEPTIONS_CONNECTIONBROKEN

Do obiektu ExceptionListenersą wysyłane tylko wyjątki wskazujące zerwane połączenie. Wszelkie inne wyjątki występujące podczas przetwarzania asynchronicznego nie są zgłaszane do obiektu ExceptionListener, dlatego aplikacja nie jest informowana o tych wyjątkach. Jest to wartość domyślna z pliku IBM MQ 8.0.0 Fix Pack 2. Patrz sekcja JMS: zmiany obiektu nasłuchiwania wyjątków w produkcie IBM MQ 8.0.

Tabela 870. Wyjątki wskazujące zerwane połączenie: środowiska i pokrewne stałe nazwy

Środowisko	Wartość
JMS narzędzie administracyjne	POŁĄCZENIE PRZERWANE
Zautomatyzowane	WMQCONSTANTS.ASYNC_EXCEPTIONS_CONNECTIONBROKEN = 1
IBM MQ Explorer	Zerwane połączenie

Zdefiniowana jest następująca dodatkowa stała:

- W systemie IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_CONNECTIONBROKEN
- Przed IBM MQ 8.0.0 Fix Pack 2: WMQCONSTANTS.ASYNC_EXCEPTIONS_DEFAULT = ASYNC_EXCEPTIONS_ALL

Pojęcia pokrewne

Wyjątki w IBM MQ classes for JMS

V 9.3.4 BALOPCJE

Steruje sposobem równoważenia aplikacji IBM MQ classes for JMS korzystających z transportu klienta w jednolitych klastrach.

Odpowiednie obiekty

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`.

Długa nazwa narzędzia administracyjnego JMS : **BALOPTIONS**

Skrócona nazwa narzędzia administracyjnego JMS : **OPTIONS**

Dostęp programowy

Procedury ustawiające/procedury pobierające

- `MQConnectionFactory.setBalancingOptions()`
- `MQConnectionFactory.getBalancingOptions()`

Wartości

IGNNONE (brak)
Stosowana jest normalna obsługa transakcji, a aplikacje nie są proszone o przeniesienie podczas transakcji.

Ta wartość jest odwzorowywana na opcję IBM MQ *BalancingOption* `MQBNO_OPTIONS_NONE`.

TRANS_IGNTRANS
Można zażądać przeniesienia aplikacji podczas transakcji.

Ta wartość jest odwzorowywana na IBM MQ *BalancingOption* `MQBNO_OPTIONS_IGNORE_TRANS`.

V 9.3.4 TYP BALTYPE

Określa, w jaki sposób aplikacje IBM MQ classes for JMS używające transportu klienta mogą być równoważone w jednolitym klastrze.

Odpowiednie obiekty

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`.

Długa nazwa narzędzia administracyjnego JMS : **BALTYPE**

Skrócona nazwa narzędzia administracyjnego JMS : **TYPE**

Dostęp programowy

Procedury ustawiające/procedury pobierające

- `MQConnectionFactory.setBalancingApplicationType()`
- `MQConnectionFactory.getBalancingApplicationType()`

Wartości

Proste
Zastosowanie ma domyślna obsługa aplikacji w jednolitym klastrze.

Ta wartość jest odwzorowywana na opcję IBM MQ *BalancingOption* `MQBNO_BALTYPE_SIMPLE`.

ŻĄDANIE_ODPOWIEDŹ
Żądanie ponownego nawiązania połączenia przez aplikację nie zostanie wysłane, jeśli serwer **MQPUT** nie został zrównoważony przez serwer **MQGET**, chyba że upłynął limit czasu.

Ta wartość jest odwzorowywana na wartość IBM MQ *BalancingOption* `MQBNO_BALTYPE_REQREP`.

V 9.3.4 LIMIT_CZASU

Określa, w jaki sposób aplikacje IBM MQ classes for JMS używające transportu klienta są równoważone w jednolitym klastrze.

Odpowiednie obiekty

`ConnectionFactory`, `QueueConnectionFactory`, `TopicConnectionFactory`, `XAConnectionFactory`, `XAQueueConnectionFactory`, `XATopicConnectionFactory`.

Długa nazwa narzędzia administracyjnego JMS : **BALTIMEOUT**

Skrócona nazwa narzędzia administracyjnego JMS : **TIMEOUT**

Dostęp programowy

Procedury ustawiające/procedury pobierające

- `MQConnectionFactory.setBalancingTimeout()`
- `MQConnectionFactory.getBalancingTimeout()`

Wartości

Nigdy
Aplikacja nigdy nie przekracza limitu czasu na potrzeby równoważenia w jednolitym klastrze.

Ta wartość jest odwzorowywana na opcję IBM MQ *BalancingOption* `MQBNO_TIMEOUT_NEVER`.

Immediate (Natychmiast)
Aplikacja natychmiast przekracza limit czasu na potrzeby równoważenia w jednolitym klastrze.

Ta wartość jest odwzorowywana na opcję IBM MQ *BalancingOption* `MQBNO_TIMEOUT_IMMEDIATE`.

DOMYŚLNA
Aplikacja przekracza limit czasu na potrzeby równoważenia w jednolitym klastrze po upływie domyślnego okresu 10 sekund.

Ta wartość jest odwzorowywana na wartość opcji IBM MQ *BalancingOption* `MQBNO_TIMEOUT_AS_DEFAULT`.

nn
Aplikacja przekracza limit czasu na potrzeby równoważenia w jednolitym klastrze po okresie *nn* sekund.

Wartość *nn* może być z zakresu od 1 do 9999999999.

BROKERCCDURSUBQ

Nazwa kolejki, z której pobierane są komunikaty subskrypcji trwałej dla ConnectionConsumer.

Odpowiednie obiekty

Temat

Długa nazwa narzędzia administracyjnego JMS : BROKERCCDURSUBQ

Skrócona nazwa narzędzia administracyjnego JMS : CCDSUB

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQTopic.setBrokerCCDurSubQueue()
- MQTopic.getBrokerCCDurSubQueue()

Wartości

SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE
Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERCCSUBQ

Nazwa kolejki, z której pobierane są komunikaty nietrwałej subskrypcji dla obiektu ConnectionConsumer.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : BROKERCCSUBQ

Skrócona nazwa narzędzia administracyjnego JMS : CCSUB

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setBrokerCCSubQueue()
- MQConnectionFactory.getBrokerCCSubQueue()

Wartości

SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERCONQ

Nazwa kolejki sterującej brokera.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : BROKERCONQ

Skrócona nazwa narzędzia administracyjnego JMS : BCON

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setBrokerControlQueue()
- MQConnectionFactory.getBrokerControlQueue()

Wartości

SYSTEM.BROKER.CONTROL.QUEUE
Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERDURSUBQ

Jeśli plik IBM MQ classes for JMS jest używany w trybie migracji dostawcy przesyłania komunikatów produktu IBM MQ , ta właściwość określa nazwę kolejki, z której są pobierane komunikaty subskrypcji trwałej.

Odpowiednie obiekty

Temat

Długa nazwa narzędzia administracyjnego JMS : BROKERDURSUBQ

Skrócona nazwa narzędzia administracyjnego JMS : BDSUB

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQTopic.setBrokerDurSubQueue()
- MQTopic.getBrokerDurSubQueue()

Wartości

SYSTEM.JMS.D.SUBSCRIBER.QUEUE
Jest to wartość domyślna.

Dowolny poprawny łańcuch
Począwszy od SYSTEM.JMS.D

Zadania pokrewne
Konfigurowanie właściwości JMS **PROVIDERVERSION**

BROKERPUBQ

Nazwa kolejki, do której wysyłane są opublikowane komunikaty (kolejka strumienia).

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, Temat, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : BROKERPUBQ

Skrócona nazwa narzędzia administracyjnego JMS : BPUB

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setBrokerPubQueue
- MQConnectionFactory.getBrokerPubQueue

Wartości

SYSTEM.BROKER.DEFAULT.STREAM
Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERPUBQMGR

Nazwa menedżera kolejek będącego właścicielem kolejki, do której są wysyłane komunikaty opublikowane w temacie.

Odpowiednie obiekty

Temat

Długa nazwa narzędzia administracyjnego JMS : BROKERPUBQMGR

Skrócona nazwa narzędzia administracyjnego JMS : BPQM

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQTopic.setBrokerPubQueueManager()
- MQTopic.getBrokerPubQueueManager()

Wartości

null
Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERQMGR

Nazwa menedżera kolejek, w którym działa broker.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : BROKERQMGR

Skrócona nazwa narzędzia administracyjnego JMS : BQM

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setBrokerQueueManager()
- MQConnectionFactory.getBrokerQueueManager()

Wartości

null
Jest to wartość domyślna.

Dowolny poprawny łańcuch

BROKERSUBQ

Jeśli plik IBM MQ classes for JMS jest używany w trybie migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ , ta właściwość określa nazwę kolejki, z której są pobierane komunikaty nietrwałej subskrypcji.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : BROKERSUBQ

Skrócona nazwa narzędzia administracyjnego JMS : BSUB

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setBrokerSubQueue()
- MQConnectionFactory.getBrokerSubQueue()

Wartości

SYSTEM.JMS.ND.SUBSCRIBER.QUEUE
Jest to wartość domyślna.

Dowolny poprawny łańcuch
Począwszy od SYSTEM.JMS.ND

Zadania pokrewne
Konfigurowanie właściwości JMS **PROVIDERVERSION**

BROKERVER

Wersja używanego brokera.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, Temat, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : BROKERVER

Skrócona nazwa narzędzia administracyjnego JMS : BVER

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setBroker()
- MQConnectionFactory.getBroker()

Wartości

V1
Aby użyć brokera publikowania/subskrypcji produktu IBM MQ lub brokera produktu IBM MQ Integrator WebSphere Event Broker, brokera zdarzeń produktu WebSphere Business Integration lub brokera komunikatów produktu WebSphere Business Integration w trybie zgodności. Jest to wartość domyślna, jeśli dla parametru TRANSPORT ustawiono wartość BIND lub CLIENT.

V2
Aby użyć brokera produktu IBM MQ Integrator, WebSphere Event Broker, WebSphere Business Integration Event Broker lub WebSphere Business Integration Message Broker w trybie rodzimym. Jest to wartość domyślna, jeśli TRANSPORT ma wartość DIRECT lub DIRECTHTTP.

nieokreślona
Po przeprowadzeniu migracji brokera z wersji V6 do wersji V7należy ustawić tę właściwość, aby nagłówki RFH2 nie były już używane. Po migracji ta właściwość nie jest już istotna.

CCDTURL

Adres URL (Uniform Resource Locator), który identyfikuje nazwę i położenie pliku zawierającego tabelę definicji kanału klienta i określa sposób uzyskiwania dostępu do tego pliku.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CCDTURL

Skrócona nazwa narzędzia administracyjnego JMS : CCDT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setCCDTURL()
- MQConnectionFactory.getCCDTURL()

Wartości

null
Jest to wartość domyślna.

Adres URL (URL)

CCSID

W przypadku fabryk połączeń ta właściwość określa identyfikator kodowanego zestawu znaków (CCSID), który ma być używany dla wewnętrznych przepływów danych z menedżerem kolejek. W przypadku miejsc docelowych właściwość definiuje identyfikator CCSID, który ma być używany do kodowania danych łańcuchowych w komunikatach MapMessages, StreamMessagesi TextMessages umieszczanych w tym miejscu docelowym.

Uwaga: Zwykle nie jest konieczna zmiana tej właściwości dla fabryk połączeń.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, Kolejka, Temat, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CCSID

Skrócona nazwa narzędzia administracyjnego JMS : CCS

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setCCSID()
- MQConnectionFactory.getCCSID()

Wartości

819
Wartość domyślna fabryki połączeń.

1208
Wartość domyślna dla miejsca docelowego.

Dowolna dodatnia liczba całkowita

Pojęcia pokrewne
JMS Konwersja komunikatu

CHANNEL

Nazwa używanego kanału połączenia klienckiego.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CHANNEL

Skrócona nazwa narzędzia administracyjnego JMS : CHAN

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setChannel()
- MQConnectionFactory.getChannel()

Wartości

SYSTEM.DEF.SVRCONN
Jest to wartość domyślna.

Dowolny poprawny łańcuch

CLEANUP

Poziom czyszczenia dla składnic subskrypcji BROKER lub MIGRATE.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLEANUP

Skrócona nazwa narzędzia administracyjnego JMS : CL

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setCleanupLevel ()
- MQConnectionFactory.getCleanupLevel ()

Wartości

Bezpieczne
Użyj bezpiecznego czyszczenia. Jest to wartość domyślna.

ASPROP,
Zgodnie z właściwością ustawioną w wierszu komend Java należy użyć procedury czyszczącej, która jest bezpieczna, silna lub nie jest wykonywana.

Brak
Nie używaj procedury czyszczącej.

silny
Użyj silnego czyszczenia.

CLEANUPINT

Odstęp czasu (w milisekundach) między kolejnymi wykonaniami w tle programu narzędziowego procedury czyszczącej publikowania/subskrypcji.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLEANUPINT

Skrócona nazwa narzędzia administracyjnego JMS : CLINT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setCleanupInterwał ()
- MQConnectionFactory.getCleanupInterwał ()

Wartości

3600000
Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

LISTA NAZW POŁĄCZEŃ

Lista nazw połączeń TCP/IP. Lista jest podejmowana w kolejności, po jednej dla każdej próby ponownego nawiązania połączenia.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : CONNECTIONNAMELIST

Skrócona nazwa narzędzia administracyjnego JMS : CNLIST

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setconnectionNameLista ()
- MQConnectionFactory.getconnectionNameList ()

Wartości

Lista oddzielonych przecinkami nazw hostów (PORT). HOSTNAME może być nazwą DNS lub adresem IP.

Domyślny PORT to 1414.

CLIENTRECONNECTOPTIONS

Opcje zarządzania ponownym połączeniem.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLIENTRECONNECTOPTIONS

Skrócona nazwa narzędzia administracyjnego JMS : CROPT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setClientReconnectOptions()
- MQConnectionFactory.getClientReconnectOptions()

Wartości

QMGR

Aplikacja może ponownie nawiązać połączenie z tym samym menedżerem kolejek, z którym była pierwotnie połączona.

Błąd o kodzie przyczyny `MQRC_RECONNECT_QMID_MISMATCH` jest zwracany, jeśli menedżer kolejek, z którym aplikacja próbuje się połączyć, określony na liście nazw połączeń, ma inny identyfikator QMID niż menedżer kolejek, z którym pierwotnie nawiązała połączenie.

Tej wartości należy użyć, jeśli można ponownie nawiązać połączenie z aplikacją, ale istnieje powinowactwo między aplikacją IBM MQ classes for JMS a menedżerem kolejek, z którym aplikacja ta nawiąże pierwsze połączenie.

Wybierz tę wartość, jeśli aplikacja ma automatycznie ponownie nawiązać połączenie z instancją rezerwową menedżera kolejek o wysokiej dostępności.

Aby użyć tej wartości programowo, należy użyć stałej WMQConstants.WMQ_CLIENT_RECONNECT_Q_MGR.

ANY

Aplikacja może ponownie nawiązać połączenie z dowolnym menedżerem kolejek określonym na liście nazw połączeń.

Opcji ponownego połączenia należy używać tylko wtedy, gdy nie ma powinowactwa między klasami IBM MQ dla aplikacji JMS i menedżerem kolejek, z którym początkowo nawiązano połączenie.

Aby użyć tej wartości z poziomu programu, należy użyć stałej WMQConstants.WMQ_CLIENT_RECONNECT.

WYŁĄCZONE

Połączenie aplikacji nie będzie ponownie nawiązywane.

Aby użyć tej wartości programowo, należy użyć stałej WMQConstants.WMQ_CLIENT_RECONNECT_DISABLED.

ASDEF

To, czy aplikacja będzie ponownie łączyć się automatycznie, zależy od wartości atrybutu kanału IBM MQ DefReconnect.

Jest to wartość domyślna.

Aby użyć tej wartości z poziomu programu, należy użyć stałej WMQConstants.WMQ_CLIENT_RECONNECT_AS_DEF.

CLIENTRECONNECTTIMEOUT

Czas do zakończenia ponownych prób nawiązania połączenia.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLIENTRECONNECTTIMEOUT

Skrócona nazwa narzędzia administracyjnego JMS : CRT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setClientReconnectTimeout()
- MQConnectionFactory.setClientReconnectTimeout()

Wartości

Odstęp czasu w sekundach. Wartość domyślna to 1800 (30 minut).

CLIENTID

Identyfikator klienta służy do jednoznacznej identyfikacji połączenia aplikacji dla subskrypcji stałych.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLIENTID

Skrócona nazwa narzędzia administracyjnego JMS : CID

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setClientId ()
- MQConnectionFactory.getClientId ()

Wartości

null
Jest to wartość domyślna.

Dowolny poprawny łańcuch

CLONESUPP

Określa, czy dwie lub więcej instancji tego samego trwałego subskrybenta tematu może działać równocześnie.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CLONESUPP

Skrócona nazwa narzędzia administracyjnego JMS : CLS

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setCloneSupport ()
- MQConnectionFactory.getCloneSupport ()

Wartości

WYŁĄCZONE
W danym momencie może działać tylko jedna instancja trwałego subskrybenta tematów. Jest to wartość domyślna.

WŁĄCZONY
Dwie lub więcej instancji tego samego stałego subskrybenta tematów może działać równocześnie, ale każda instancja musi działać w oddzielnej wirtualnej maszynie języka Java (JVM).

COMPHDR

Lista technik, których można użyć do kompresji danych nagłówka w połączeniu.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : COMPHDR

Skrócona nazwa narzędzia administracyjnego JMS : HC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setHdrCompList()
- MQConnectionFactory.getHdrCompList()

Wartości

Brak
Jest to wartość domyślna.

SYSTEM
Kompresja nagłówka komunikatu RLE jest wykonywana.

COMPMSG

Lista technik, których można użyć do kompresji danych komunikatu w połączeniu.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : COMPMSG

Skrócona nazwa narzędzia administracyjnego JMS : MC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMsgCompList()
- MQConnectionFactory.getMsgCompList()

Wartości

Brak
Jest to wartość domyślna.

Lista zawierająca jedną lub więcej następujących wartości rozdzielonych znakami odstępu:
RLE ZLIBFAST ZLIBHIGH

CONNOPT

Steruje sposobem, w jaki aplikacje produktu IBM MQ classes for JMS używające transportu powiązań łączą się z menedżerem kolejek.

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory, XATopicConnection.

Długa nazwa narzędzia administracyjnego JMS : CONNOPT

Skrócona nazwa narzędzia administracyjnego JMS : CNOPT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMQConnectionOpcje ()
- MQConnectionFactory.getMQConnectionOptions ()

Wartości

STANDARDOWA
Rodzaj powiązania między aplikacją i menedżerem kolejek zależy od wartości atrybutu *DefaultBindType* menedżera kolejek. Wartość STANDARD jest odwzorowywana na wartość IBM MQ *ConnectOption* MQCNO_STANDARD_BINDING.

WSPÓŁUŻYTKOWANY
Aplikacja i agent lokalnego menedżera kolejek działają w oddzielnych jednostkach wykonywania, ale współużytkują pewne zasoby. Ta wartość jest odwzorowywana na opcję IBM MQ *ConnectOption* MQCNO_SHARED_BINDING.

Odizolowane
Aplikacja i agent lokalnego menedżera kolejek działają w oddzielnych jednostkach wykonywania i nie współużytkują żadnych zasobów. Wartość IZOLOWANA jest odwzorowywana na opcję IBM MQ *ConnectOption* MQCNO_ISOLATED_BINDING.

Krótka ścieżka
Aplikacja i agent lokalnego menedżera kolejek działają w tej samej jednostce wykonywania. Ta wartość jest odwzorowywana na opcję IBM MQ *ConnectOption* MQCNO_FASTPATH_BINDING.

SERIALQM,
Aplikacja żąda wyłącznego użycia znacznika połączenia w zasięgu menedżera kolejek. Ta wartość jest odwzorowywana na parametr IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_Q_MGR.

SERIALQSG
Aplikacja żąda wyłącznego użycia znacznika połączenia w obrębie grupy współużytkowania kolejek, do której należy menedżer kolejek. Wartość SERIALQSG jest odwzorowywana na opcję IBM MQ *ConnectOption* MQCNO_SERIALIZE_CONN_TAG_QSG.

OGRANICZONE ZARZĄDZANIE JAKOŚCIĄ
Aplikacja żąda współużytkowanego użycia znacznika połączenia, ale istnieją ograniczenia dotyczące współużytkowanego użycia znacznika połączenia w zasięgu menedżera kolejek. Ta wartość jest odwzorowywana na IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_Q_MGR.

Ograniczone QSG
Aplikacja żąda współużytkowanego użycia znacznika połączenia, ale istnieją ograniczenia dotyczące współużytkowanego użycia znacznika połączenia w zasięgu grupy współużytkowania kolejek, do której należy menedżer kolejek. Ta wartość jest odwzorowywana na opcję IBM MQ *ConnectOption* MQCNO_RESTRICT_CONN_TAG_QSG.

Więcej informacji na temat opcji połączeń produktu IBM MQ zawiera sekcja Nawiązywanie połączenia z menedżerem kolejek przy użyciu wywołania MQCONNX.

CONNTAG

Znacznik, który menedżer kolejek przypisuje do zasobów aktualizowanych przez aplikację w jednostce pracy, gdy aplikacja jest połączona z menedżerem kolejek.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : CONNTAG

Skrócona nazwa narzędzia administracyjnego JMS : CNTAG

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setConnZnacznik ()
- MQConnectionFactory.getConnTag ()

Wartości

Tablica bajtów zawierająca 128 elementów, przy czym każdy element ma wartość 0
Jest to wartość domyślna.

Dowolny łańcuch
Wartość jest obcinana, jeśli jest dłuższa niż 128 bajtów.

OPIS

Opis zapisanego obiektu.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, Kolejka, Temat, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : DESCRIPTION

Skrócona nazwa narzędzia administracyjnego JMS : DESC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setDescription()
- MQConnectionFactory.getDescription()

Wartości

null
Jest to wartość domyślna.

Dowolny poprawny łańcuch
`

DIRECTAUTH

Określa, czy uwierzytelnianie TLS jest używane w połączeniu z brokerem w czasie rzeczywistym.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : DIRECTAUTH

Skrócona nazwa narzędzia administracyjnego JMS : DAUTH

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setDirectAuth ()
- MQConnectionFactory.getDirectAuth ()

Wartości

PODSTAWOWY
Brak uwierzytelniania, uwierzytelnianie nazwy użytkownika lub uwierzytelnianie hasła. Jest to wartość domyślna.

Certyfikat
Uwierzytelnianie przy użyciu certyfikatu klucza publicznego.

ENCODING

Sposób reprezentowania danych liczbowych w treści komunikatu, gdy komunikat jest wysyłany do tego miejsca docelowego. Ta właściwość określa reprezentację binarnych liczb całkowitych, upakowanych liczb dziesiętnych i liczb zmiennopozycyjnych.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : ENCODING

Skrócona nazwa narzędzia administracyjnego JMS : ENC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setEncoding()
- MQDestination.getEncoding()

Wartości

Właściwość ENCODING

Poprawne wartości, które może przyjąć właściwość `ENCODING` , są tworzone na podstawie trzech podwłaściwości:

Kodowanie na podstawie liczb całkowitych
Normalny lub odwrócony

Kodowanie dziesiętne
Normalny lub odwrócony

kodowanie zmiennopozycyjne
Normalny IEEE, odwrócony IEEE lub z/OS

Właściwość ENCODING jest wyrażona w postaci łańcucha trzyznakowego z następującą składnią:

```
{N|R}{N|R}{N|R|3}
```

W tym łańcuchu:

- N oznacza normalne
- R oznacza odwrócone
- 3 oznacza z/OS
- Pierwszy znak reprezentuje *kodowanie całkowite*
- Drugi znak reprezentuje *kodowanie dziesiętne*
- Trzeci znak reprezentuje *kodowanie zmiennopozycyjne*

W ten sposób udostępniono zestaw dwunastu możliwych wartości właściwości ENCODING .

Istnieje dodatkowa wartość, łańcuch NATIVE, który ustawia odpowiednie wartości kodowania dla platformy Java .

W poniższych przykładach przedstawiono poprawne kombinacje parametrów ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

EXPIRY

Czas, po którym komunikaty w miejscu docelowym tracą ważność.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : TERMIN WAŻNOŚCI

Skrócona nazwa narzędzia administracyjnego JMS : EXP

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setExpiry()
- MQDestination.getExpiry()

Wartości

APP
Utrata ważności może być zdefiniowana przez aplikację JMS . Jest to wartość domyślna.

ULIM
Nie występuje utrata ważności.

0
Nie występuje utrata ważności.

Dowolna dodatnia liczba całkowita oznaczająca utratę ważności w milisekundach.

FAILIFQUIESCE

Ta właściwość określa, czy wywołania niektórych metod nie powiodą się, jeśli menedżer kolejek jest w stanie wyciszania, czy aplikacja nawiązuje połączenie z menedżerem kolejek przy użyciu transportu

CLIENT, a kanał używany przez aplikację został przełączony w stan wyciszania, na przykład za pomocą komend **STOP CHANNEL** lub **STOP CHANNEL MODE(QUIESCE)** MQSC.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, Kolejka, Temat, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : FAILIFQUIESCE

Skrócona nazwa narzędzia administracyjnego JMS : FIQ

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setFailIfQuiesce()
- MQConnectionFactory.getFailIfQuiesce()

Wartości

YES
Wywołania niektórych metod kończą się niepowodzeniem, jeśli menedżer kolejek jest w stanie wyciszania lub kanał używany do nawiązywania połączenia z menedżerem kolejek jest w stanie wyciszania. Jeśli aplikacja wykryje jeden z tych warunków, może zakończyć swoje natychmiastowe zadanie i zamknąć połączenie, umożliwiając zatrzymanie menedżera kolejek lub instancji kanału. Jest to wartość domyślna.

NO
Wywołanie metody nie powiodło się, ponieważ menedżer kolejek lub kanał używany do nawiązywania połączenia z menedżerem kolejek jest w stanie wyciszania. Jeśli ta wartość zostanie określona, aplikacja nie może wykryć, że menedżer kolejek lub kanał jest wyciszany. Aplikacja może kontynuować wykonywanie operacji na menedżerze kolejek, co uniemożliwi zatrzymanie menedżera kolejek.

HOSTNAME

W przypadku połączenia z menedżerem kolejek jest to nazwa hosta lub adres IP systemu, w którym działa menedżer kolejek, lub, w przypadku połączenia w czasie rzeczywistym z brokerem, nazwa hosta lub adres IP systemu, w którym działa broker.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : HOSTNAME

Skrócona nazwa narzędzia administracyjnego JMS : HOST

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setHostNazwa ()
- MQConnectionFactory.getHostNazwa ()

Wartości

localhost
Jest to wartość domyślna.

Dowolny poprawny łańcuch

LOCALADDRESS

W przypadku połączenia z menedżerem kolejek ta właściwość określa lokalny interfejs sieciowy, który ma być używany, port lokalny lub zakres portów lokalnych, które mają być używane.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : LOCALADDRESS

Skrócona nazwa narzędzia administracyjnego JMS : LA

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setLocalAddress ()
- MQConnectionFactory.getLocalAddress ()

Wartości

"" (pusty łańcuch)
Jest to wartość domyślna.

Łańcuch w formacie [adres-IP] [(port-niski [, port-wysoki])]
Poniżej przedstawiono kilka przykładów:

`192.0.2.0`

Kanał zostanie lokalnie powiązany z adresem 192.0.2.0 .

`192.0.2.0(1000)`

Kanał jest lokalnie powiązany z adresem 192.0.2.0 i używa portu 1000.

`192.0.2.0(1000,2000)`

Kanał jest lokalnie powiązany z adresem 192.0.2.0 i używa portu z zakresu od 1000 do 2000.

`(1000)`

Kanał zostanie lokalnie powiązany z portem 1000.

`(1000,2000)`

Kanał jest lokalnie powiązany z portem z zakresu od 1000 do 2000.

Zamiast adresu IP można podać nazwę hosta. W przypadku połączenia w czasie rzeczywistym z brokerem ta właściwość ma zastosowanie tylko wtedy, gdy używane jest rozsyłanie grupowe, a wartość właściwości nie może zawierać numeru portu ani zakresu numerów portów. W tym przypadku jedynymi poprawnymi wartościami właściwości są: null, adres IP lub nazwa hosta.

STYL NAZWY MAPY

Umożliwia użycie stylu zgodności dla nazw elementów MapMessage .

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : MAPNAMESTYLE

Skrócona nazwa narzędzia administracyjnego JMS : MNST

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMapNameStyle()
- MQConnectionFactory.getMapNameStyle()

Wartości

STANDARDOWA
Zostanie użyty standardowy format nazewnictwa elementów com.ibm.jms.JMSMapMessage . Jest to wartość domyślna, która umożliwia użycie jako nazwy elementu identyfikatorów Java , które nie są dozwolone.

Kompatybilny
Zostanie użyty starszy format nazewnictwa elementów com.ibm.jms.JMSMapMessage . Jako nazwy elementu mogą być używane tylko dozwolone identyfikatory Java . Jest to wymagane tylko wtedy, gdy komunikaty odwzorowania są wysyłane do aplikacji korzystającej z systemu IBM MQ classes for JMS w wersji wcześniejszej niż 5.3.

MAXBUFFSIZE

Maksymalna liczba odebranych komunikatów, które mogą być przechowywane w wewnętrznym buforze komunikatów podczas oczekiwania na przetworzenie przez aplikację. Ta właściwość ma zastosowanie tylko wtedy, gdy TRANSPORT ma wartość DIRECT lub DIRECTHTTP.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : MAXBUFFSIZE

Skrócona nazwa narzędzia administracyjnego JMS : MBSZ

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMaxBufferSize()
- MQConnectionFactory.getMaxBufferSize()

Wartości

1000
Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

MDREAD

Ta właściwość określa, czy aplikacja JMS może wyodrębniać wartości pól MQMD.

Odpowiednie obiekty

Długa nazwa narzędzia administracyjnego JMS : MDREAD

Skrócona nazwa narzędzia administracyjnego JMS : MDR

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setMQMDReadEnabled()
- MQDestination.getMQMDReadEnabled()

Wartości

NO
Podczas wysyłania komunikatów właściwości JMS_IBM_MQMD* wysłanego komunikatu nie są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD. Podczas odbierania komunikatów żadna właściwość JMS_IBM_MQMD* nie jest dostępna w odebranym komunikacie, nawet jeśli nadawca ustawił niektóre lub wszystkie z nich. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości False.

Tak
Podczas wysyłania komunikatów wszystkie właściwości JMS_IBM_MQMD* wysłanego komunikatu są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD, w tym także właściwości, które nie zostały jawnie ustawione przez nadawcę. Podczas odbierania komunikatów wszystkie właściwości JMS_IBM_MQMD* są dostępne w odebranym komunikacie, w tym właściwości, które nie zostały jawnie ustawione przez nadawcę.

W przypadku programów należy użyć wartości True.

MDWRITE

Ta właściwość określa, czy aplikacja JMS może ustawiać wartości pól MQMD.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : MDWRITE

Skrócona nazwa narzędzia administracyjnego JMS : MDR

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setMQMDWriteEnabled()
- MQDestination.getMQMDWriteEnabled()

Wartości

NO
Wszystkie właściwości JMS_IBM_MQMD* są ignorowane, a ich wartości nie są kopiowane do bazowej struktury MQMD. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości False.

YES
Właściwości JMS_IBM_MQMD* są przetwarzane. Ich wartości są kopiowane do bazowej struktury MQMD.

W przypadku programów należy użyć wartości True.

MDMSGCTX

Poziom kontekstu komunikatu, który ma zostać ustawiony przez aplikację JMS . Aby ta właściwość miała zastosowanie, aplikacja musi być uruchamiana z odpowiednimi uprawnieniami dotyczącymi kontekstu.

Odpowiednie obiekty

Długa nazwa narzędzia administracyjnego JMS : MDMSGCTX

Skrócona nazwa narzędzia administracyjnego JMS : MDCTX

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setMQMDMessageContext()
- MQDestination.getMQMDMessageContext()

Wartości

DOMYŚLNA
Wywołanie funkcji API MQOPEN i struktura MQPMO nie określają jawnych opcji kontekstu komunikatu. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQ_MDCTX_DEFAULT.

SET_IDENTITY_CONTEXT,
Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_IDENTITY_CONTEXT, a struktura MQPMO określa opcję MQPMO_SET_IDENTITY_CONTEXT.

W przypadku programów należy użyć wartości WMQ_MDCTX_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT
Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_ALL_CONTEXT, a struktura MQPMO określa opcję MQPMO_SET_ALL_CONTEXT.

W przypadku programów należy użyć wartości WMQ_MDCTX_SET_ALL_CONTEXT.

MSGBATCHSZ

Maksymalna liczba komunikatów, które można pobrać z kolejki w jednym pakiecie podczas korzystania z asynchronicznego dostarczania komunikatów.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : MAXBUFFSIZE

Skrócona nazwa narzędzia administracyjnego JMS : MBSZ

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMsgBatchSize()
- MQConnectionFactory.getMsgBatchSize()

Wartości

10
Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

MSGBODY

Określa, czy aplikacja JMS uzyskuje dostęp do MQRFH2 komunikatu IBM MQ w ramach ładunku komunikatu.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : WMQ_MESSAGE_BODY

Skrócona nazwa narzędzia administracyjnego JMS : MBODY

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMessageBodyStyle()
- MQConnectionFactory.getMessageBodyStyle()

Wartości

Nieokreślone
Podczas wysyłania produkt IBM MQ classes for JMS generuje lub nie generuje i dołącza nagłówek MQRFH2 , w zależności od wartości właściwości WMQ_TARGET_CLIENT. Podczas odbierania działa jako wartość JMS.

JMS
Podczas wysyłania produkt IBM MQ classes for JMS automatycznie generuje nagłówek MQRFH2 i dołącza go do komunikatu IBM MQ .

Podczas odbierania produkt IBM MQ classes for JMS ustawia właściwości komunikatu JMS zgodnie z wartościami w nagłówku MQRFH2 (jeśli istnieje). Nie przedstawia on nagłówka MQRFH2 jako części treści komunikatu JMS .

MQ
Podczas wysyłania produkt IBM MQ classes for JMS nie generuje nagłówka MQRFH2.

Podczas odbierania produkt IBM MQ classes for JMS prezentuje MQRFH2 jako część treści komunikatu JMS .

MSGRETENTION

Określa, czy konsument połączenia przechowuje niedostarczone komunikaty w kolejce wejściowej.

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory,

Długa nazwa narzędzia administracyjnego JMS : MSGRERYWALIZACJA

Skrócona nazwa narzędzia administracyjnego JMS : MRET

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMessageCzas przechowywania ()
- MQConnectionFactory.getMessageRetention ()

Wartości

Tak
Niedostarczone komunikaty pozostają w kolejce wejściowej. Jest to wartość domyślna.

Nie
Niedostarczone komunikaty są przetwarzane zgodnie z ich opcjami dyspozycji.

MSGSELECTION

Określa, czy wybór komunikatów jest dokonywany przez IBM MQ classes for JMS , czy przez broker. Jeśli parametr TRANSPORT ma wartość DIRECT, wybór komunikatów jest zawsze dokonywany przez broker, a wartość parametru MSGSELECTION jest ignorowana. Wybór komunikatów przez broker nie jest obsługiwany, jeśli BROKERVER ma wartość V1.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : MSGSELECTION

Skrócona nazwa narzędzia administracyjnego JMS : MSEL

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMessageWybór ()
- MQConnectionFactory.getMessage-wybór ()

Wartości

KLIENT
Wybór komunikatów jest dokonywany przez IBM MQ classes for JMS. Jest to wartość domyślna.

BROKER
Wybór komunikatów jest dokonywany przez broker.

MULTICAST

Aby włączyć rozsyłanie grupowe w czasie rzeczywistym dla połączenia z brokerem oraz, jeśli jest włączone, aby określić dokładny sposób, w jaki rozsyłanie grupowe jest używane do dostarczania komunikatów z brokera do konsumenta komunikatów. Ta właściwość nie ma wpływu na sposób, w jaki producent komunikatów wysyła komunikaty do brokera.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, Temat

Długa nazwa narzędzia administracyjnego JMS : MULTICAST

Skrócona nazwa narzędzia administracyjnego JMS : MCAST

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setMulticast()
- MQConnectionFactory.getMulticast()

Wartości

WYŁĄCZONE
Komunikaty nie są dostarczane do odbiorcy komunikatów przy użyciu rozsyłania grupowego. Jest to wartość domyślna dla obiektów fabryki ConnectionFactory i TopicConnection.

FKD
Komunikaty są dostarczane do konsumenta komunikatów zgodnie z ustawieniem rozsyłania dla fabryki połączeń powiązanej z konsumentem komunikatów. Ustawienie rozsyłania grupowego dla fabryki połączeń jest odnotowywana w momencie tworzenia konsumenta komunikatów. Ta wartość jest poprawna tylko dla obiektów tematu i jest wartością domyślną dla obiektów tematu.

WŁĄCZONY
Jeśli temat jest skonfigurowany do rozsyłania grupowego w brokerze, komunikaty są dostarczane do odbiorcy komunikatów przy użyciu transportu rozsyłania grupowego. Niezawodna jakość usługi jest używana, jeśli temat jest skonfigurowany do niezawodnego rozsyłania grupowego.

Niezawodne
Jeśli temat jest skonfigurowany do niezawodnego rozsyłania grupowego w brokerze, komunikaty są dostarczane do odbiorcy komunikatów przy użyciu transportu rozsyłania grupowego z niezawodną jakością usługi. Jeśli temat nie jest skonfigurowany do niezawodnego rozsyłania grupowego, nie można utworzyć konsumenta komunikatów dla tematu.

NOTR
Jeśli temat jest skonfigurowany do rozsyłania grupowego w brokerze, komunikaty są dostarczane do odbiorcy komunikatów przy użyciu transportu rozsyłania grupowego. Niezawodna jakość usługi nie jest używana, nawet jeśli temat jest skonfigurowany do niezawodnego rozsyłania grupowego.

OPTIMISTICPUBLICATION

Ta właściwość określa, czy produkt IBM MQ classes for JMS natychmiast zwraca sterowanie do publikatora, który opublikował komunikat, czy też zwraca sterowanie tylko po zakończeniu przetwarzania związanego z wywołaniem i może zgłosić wynik do publikatora.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Pełna nazwa narzędzia administracyjnego JMS : OPTIMISTICPUBLIKOWANIE

Skrócona nazwa narzędzia administracyjnego JMS : OPTPUB

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setOptimisticPublication ()
- MQConnectionFactory.getOptimisticPublication ()

Wartości

NO
Gdy publikator publikuje komunikat, IBM MQ classes for JMS nie zwraca kontroli do publikatora, dopóki nie zakończy przetwarzania związanego z wywołaniem i nie będzie mógł zgłosić wyniku do publikatora. Jest to wartość domyślna.

YES
Gdy publikator publikuje komunikat, produkt IBM MQ classes for JMS natychmiast zwraca sterowanie do publikatora, zanim zakończy przetwarzanie powiązane z wywołaniem i będzie mógł zgłosić wynik do publikatora. Program IBM MQ classes for JMS zgłasza wynik tylko wtedy, gdy publikator zatwierdzi komunikat.

OUTCOMENOTIFICATION

Ta właściwość określa, czy produkt IBM MQ classes for JMS natychmiast zwraca kontrolę subskrybentowi, który właśnie potwierdził lub zatwierdził komunikat, czy też zwraca kontrolę tylko po zakończeniu przetwarzania związanego z wywołaniem i może zgłosić wynik subskrybentowi.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : OUTCOMENOTIFICATION

Skrócona nazwa narzędzia administracyjnego JMS : NOTIFY

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setOutcomeNotification ()
- MQConnectionFactory.getOutcomeNotification ()

Wartości

YES
Gdy subskrybent potwierdzi lub zatwierdzi komunikat, produkt IBM MQ classes for JMS nie zwraca kontroli do subskrybenta, dopóki nie zakończy przetwarzania związanego z wywołaniem i nie będzie mógł zgłosić wyniku do subskrybenta. Jest to wartość domyślna.

NO
Gdy subskrybent potwierdzi lub zatwierdzi komunikat, produkt IBM MQ classes for JMS natychmiast zwraca sterowanie do subskrybenta, zanim zakończy przetwarzanie powiązane z wywołaniem i będzie mógł zgłosić wynik do subskrybenta.

PERSISTENCE

Trwałość komunikatów wysyłanych do miejsca docelowego.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : PERSISTENCE

Skrócona nazwa narzędzia administracyjnego JMS : PER

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setPersistence()
- MQDestination.getPersistence()

Wartości

APP
Trwałość jest definiowana przez aplikację JMS . Jest to wartość domyślna.

QDEF,
Trwałość przyjmuje wartość domyślną kolejki.

PERS
Komunikaty są trwałe.

NIE
Komunikaty są nietrwałe.

WYSOKI
Więcej informacji na temat użycia tej wartości można znaleźć w sekcji Komunikaty trwałe produktuJMS .

POLLINGINT

Jeśli każdy obiekt nasłuchiwania komunikatów w ramach sesji nie ma odpowiedniego komunikatu w swojej kolejce, jest to maksymalny odstęp czasu (w milisekundach), który upływa przed ponowną próbą pobrania komunikatu z kolejki przez każdy obiekt nasłuchiwania komunikatów. Jeśli często zdarza się, że dla żadnego z obiektów nasłuchiwania komunikatów w sesji nie jest dostępny odpowiedni komunikat, należy rozważyć zwiększenie wartości tej właściwości. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość TRANSPORT ma wartość BIND lub CLIENT.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : POLLINGINT

Skrócona nazwa narzędzia administracyjnego JMS : PINT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setPollingInterwał ()
- MQConnectionFactory.getPollingInterwał ()

Wartości

5000
Jest to wartość domyślna.

Dowolna dodatnia liczba całkowita

PORT

W przypadku połączenia z menedżerem kolejek jest to numer portu, na którym nasłuchuje menedżer kolejek, lub, w przypadku połączenia w czasie rzeczywistym z brokerem, numer portu, na którym broker nasłuchuje połączeń w czasie rzeczywistym.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : PORT

Skrócona nazwa narzędzia administracyjnego JMS : PORT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setPort()
- MQConnectionFactory.getPort()

Wartości

1414
Jest to wartość domyślna, jeśli dla parametru TRANSPORT ustawiono wartość CLIENT.

1506
Jest to wartość domyślna, jeśli TRANSPORT ma wartość DIRECT lub DIRECTHTTP.

Dowolna dodatnia liczba całkowita

PRIORYTET

Priorytet komunikatów wysyłanych do miejsca docelowego.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : PRIORITY

Skrócona nazwa narzędzia administracyjnego JMS : PRI

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setPriority()
- MQDestination.getPriority()

Wartości

APP
Priorytet jest definiowany przez aplikację JMS . Jest to wartość domyślna.

QDEF,
Priorytet przyjmuje wartość domyślną kolejki.

Dowolna liczba całkowita z zakresu od 0 do 9
Od najniższej do najwyższej.

PROCESSDURATION

Ta właściwość określa, czy subskrybent gwarantuje szybkie przetworzenie każdego komunikatu, który otrzyma, przed zwróceniem kontroli do produktu IBM MQ classes for JMS.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROCESSDURATION

Skrócona nazwa narzędzia administracyjnego JMS : PROCDUR

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setProcessDuration ()
- MQConnectionFactory.getProcessDuration ()

Wartości

NIEZNANY
Subskrybent nie może zagwarantować, jak szybko może przetwarzać każdy otrzymany komunikat. Jest to wartość domyślna.

Krótki
Subskrybent gwarantuje szybkie przetworzenie każdego komunikatu, który otrzyma, przed zwróceniem kontroli do IBM MQ classes for JMS.

PROVIDERVERSION

Ta właściwość odróżnia trzy tryby przesyłania komunikatów produktu IBM MQ : tryb normalny dostawcy przesyłania komunikatów produktu IBM MQ , tryb normalny dostawcy przesyłania komunikatów produktu IBM MQ z ograniczeniami i tryb migracji dostawcy przesyłania komunikatów produktu IBM MQ .

W trybie normalnym dostawcy usługi przesyłania komunikatów produktu IBM MQ wszystkie funkcje menedżera kolejek produktu IBM MQ są używane do implementowania interfejsu JMS. Ten tryb jest zoptymalizowany pod kątem używania interfejsu API produktu JMS 2.0 i jego funkcji. Tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ z ograniczeniami używa interfejsu API produktu JMS 2.0 , ale nie nowych funkcji, takich jak subskrypcje współużytkowane, opóźnione dostarczanie lub wysyłanie asynchroniczne.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection , Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROVIDERVERSION

Skrócona nazwa narzędzia administracyjnego JMS : PVER

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setProviderwersja ()
- MQConnectionFactory.getProvider()

Wartości

Właściwość **PROVIDERVERSION** można ustawić na dowolną z następujących wartości: 8 (tryb normalny), 7 (tryb normalny z ograniczeniami), 6 (tryb migracji) lub `nie określono` (wartość domyślna). Wartość podana dla właściwości **PROVIDERVERSION** musi być łańcuchem. Jeśli jest podana opcja 8, 7 lub 6, dopuszczalne są następujące formaty:

- V.R.M.F
- V.R.M
- V.R
- V

Gdzie: V, R, M i F są wartościami całkowitymi większymi niż zero lub równymi zero. Dodatkowe wartości R, M i F są opcjonalne i można ich używać, jeśli wymagana jest precyzyjna kontrola. Na przykład jeśli użytkownik chce użyć poziomu **PROVIDERVERSION** o wartości 7, może ustawić **PROVIDERVERSION**=7, `7.0`, `7.0.0` lub `7.0.0.0`.

8 – tryb normalny
Aplikacja JMS używa trybu normalnego dostawcy usługi przesyłania komunikatów produktu IBM MQ. W trybie normalnym używane są wszystkie funkcje menedżera kolejek produktu IBM MQ służące do implementowania usług JMS. Ten tryb jest zoptymalizowany pod kątem użycia funkcjonalności i interfejsu API JMS 2.0.

W przypadku nawiązywania połączenia z menedżerem kolejek przy poziomie komend 800 lub nowszym można użyć wszystkich funkcji i funkcji interfejsu API produktu JMS 2.0 , takich jak wysyłanie asynchroniczne, opóźniona dostawa lub subskrypcja współużytkowana.

Jeśli menedżer kolejek określony w ustawieniach fabryki połączeń nie jest menedżerem kolejek w wersji IBM MQ 8.0.0 lub nowszej, metoda `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSFMQ0003`.

Tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ korzysta z funkcji konwersacji współużytkowanych, a liczba konwersacji, które mogą być współużytkowane, jest kontrolowana przez właściwość **SHARECNV()** na kanale połączenia z serwerem. Jeśli w przypadku tej właściwości jest ustawiona wartość `0`, nie można używać trybu normalnego dostawcy usługi przesyłania komunikatów produktu IBM MQ, a wykonywanie metody `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSCC5007`.

7 – tryb normalny z ograniczeniami
Aplikacja JMS używa trybu normalnego z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ. W tym trybie używany jest interfejs API JMS 2.0, ale nie są używane nowe funkcje, takie jak subskrypcje współużytkowane, opóźniona dostawa i wysyłanie asynchroniczne.

Jeśli parametr **PROVIDERVERSION** zostanie ustawiony na wartość 7, to dostępny jest tylko zwykły dostawca usługi przesyłania komunikatów produktu IBM MQ z ograniczeniem trybu operacji.

Jeśli połączenie jest nawiązywane w trybie normalnym z ograniczeniami, z menedżerem kolejek o poziomie komend niższym niż 800, można użyć funkcji API JMS 2.0 , ale nie funkcji wysyłania asynchronicznego, opóźnionej dostawy lub subskrypcji współużytkowanej.

Tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ z ograniczeniami korzysta z funkcji konwersacji współużytkowanych, a liczba konwersacji, które mogą być współużytkowane, jest kontrolowana przez właściwość **SHARECNV()** na kanale połączenia z serwerem. Jeśli w przypadku tej właściwości jest ustawiona wartość `0`, nie można używać trybu normalnego z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ, a wykonywanie metody `createConnection` zakończy się niepowodzeniem i zostanie zgłoszony wyjątek `JMSCC5007`.

6 – tryb migracji
Aplikacja JMS używa trybu migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.

Przy użyciu tego trybu można nawiązać połączenie z menedżerem kolejek produktu IBM MQ 8.0lub nowszym, ale żadna z nowych funkcji menedżera kolejek produktu IBM MQ classes for JMS nie jest używana, na przykład funkcja odczytu z wyprzedzeniem lub przetwarzania strumieniowego.

Jeśli klient IBM MQ 8.0 lub nowszy nawiązuje połączenie z menedżerem kolejek produktu IBM MQ 8.0 lub nowszym, wówczas wybór komunikatów jest dokonywany przez menedżer kolejek, a nie przez system klienta.

Jeśli określono tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ i zostanie podjęta próba użycia dowolnego z interfejsów API języka JMS 2.0 , wywołanie metody interfejsu API nie powiedzie się i zostanie zgłoszony wyjątek `JMSCC5007`.

nieokreślona (wartość domyślna)
Właściwość **PROVIDERVERSION** jest domyślnie ustawiona na wartość *nie określono*.

Fabryka połączeń, która została utworzona w poprzedniej wersji produktu IBM MQ classes for JMS w interfejsie JNDI, przyjmuje tę wartość, gdy fabryka połączeń jest używana z nową wersją produktu IBM MQ classes for JMS. Do określania używanego trybu operacji używany jest poniższy algorytm. Ten algorytm jest używany w przypadku wywołania metody `createConnection`. Są w nim również używane inne aspekty fabryki połączeń w celu określenia, czy wymagany jest tryb normalny przesyłania komunikatów produktu IBM MQ, tryb normalny z ograniczeniami, czy tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.

1. Najpierw podejmowana jest próba użycia dostawcy usługi przesyłania komunikatów produktu IBM MQ.

2. Jeśli połączony jest inny menedżer kolejek niż menedżer produktu IBM MQ 8.0 lub nowszy, podejmowana jest próba użycia trybu normalnego z ograniczeniami dostawcy usługi przesyłania komunikatów produktu IBM MQ.
3. Jeśli połączony jest inny menedżer kolejek niż menedżer produktu IBM WebSphere MQ 7.0.1 lub nowszego, połączenie jest zamykane i w zamian używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.
4. Jeśli właściwość **SHARECNV** na kanale połączenia z serwerem jest ustawiona na 0, to połączenie jest zamykane, a zamiast tego używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.
5. Jeśli właściwość **BROKERVER** jest ustawiona na wartość V1 lub na wartość domyślną *unspecified* (nieokreślona), nadal używany jest tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ .

   Sekcja ALTER QMGR zawiera informacje o parametrze PSMODE komendy ALTER QMGR, w tym dokładniejsze informacje o kompatybilności.
6. Jeśli parametr **BROKERVER** jest ustawiony na wartość V2, to działanie jest zależne od wartości **BROKERQMGR**:
   - Jeśli wartość **BROKERQMGR** jest `pusta`:

     Jeśli kolejka podana we właściwości **BROKERCONQ** może zostać otwarta dla danych wyjściowych (co jest równoznaczne z powodzeniem wykonania komendy `MQOPEN` dla danych wyjściowych), a w przypadku właściwości **PSMODE** w menedżerze kolejek ustawiona jest wartość `COMPAT` lub `DISABLED`, wówczas używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.
   - Jeśli kolejka określona przez właściwość **BROKERCONQ** nie może zostać otwarta dla danych wyjściowych lub atrybut **PSMODE** jest ustawiony na wartość `WŁĄCZONE`:

     Używany jest tryb normalny dostawcy usługi przesyłania komunikatów produktu IBM MQ.
   - Jeśli wartość **BROKERQMGR** jest `niepusta`:

     Używany jest tryb migracji dostawcy usługi przesyłania komunikatów produktu IBM MQ.

Jeśli nie można zmienić używanej fabryki połączeń, można użyć właściwości `com.ibm.msg.client.wmq.overrideProviderVersion`, aby nadpisać dowolne ustawienie w fabryce połączeń. To nadpisanie ma zastosowanie do wszystkich fabryk połączeń w maszynie JVM, ale rzeczywiste obiekty fabryki połączeń nie są modyfikowane.

Zadania pokrewne

Konfigurowanie właściwości JMS **PROVIDERVERSION**

PROXYHOSTNAME

Nazwa hosta lub adres IP systemu, w którym działa serwer proxy, jeśli używane jest połączenie w czasie rzeczywistym z brokerem za pośrednictwem serwera proxy.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROXYHOSTNAME

Skrócona nazwa narzędzia administracyjnego JMS : PHOST

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setProxyHostName()
- MQConnectionFactory.getProxyHostName()

Wartości

null
Nazwa hosta serwera proxy. Jest to wartość domyślna.

PROXYPORT

Numer portu, na którym nasłuchuje serwer proxy podczas używania połączenia w czasie rzeczywistym z brokerem za pośrednictwem serwera proxy.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROXYPORT

Skrócona nazwa narzędzia administracyjnego JMS : PPORT

Dostęp programowy

Procedury ustawiające/procedury pobierające

MQConnectionFactory.setProxyPort ()

MQConnectionFactory.getProxyPort ()

Wartości

443
Numer portu serwera proxy. Jest to wartość domyślna.

PUBACKINT

Liczba komunikatów opublikowanych przez publikator, zanim produkt IBM MQ classes for JMS zażąda potwierdzenia od brokera.

Jeśli wartość tej właściwości zostanie zmniejszona, produkt IBM MQ classes for JMS będzie częściej żądać potwierdzeń, co spowoduje zmniejszenie wydajności publikatora. Jeśli ta wartość zostanie podniesiona, IBM MQ classes for JMS zgłosi wyjątek po dłuższym czasie, jeśli działanie brokera zakończy się niepowodzeniem. Ta właściwość ma zastosowanie tylko wtedy, gdy właściwość TRANSPORT ma wartość BIND lub CLIENT.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : PROXYPORT

Skrócona nazwa narzędzia administracyjnego JMS : PPORT

Dostęp programowy

Procedury ustawiające/procedury pobierające

MQConnectionFactory.setPubAckInterval()

MQConnectionFactory.getPubAckInterval()

Wartości

25
Każda dodatnia liczba całkowita może być wartością domyślną.

PUTASYNCALLOWED

Ta właściwość określa, czy producenci komunikatów mogą używać asynchronicznych operacji put w celu wysyłania komunikatów do tego miejsca docelowego.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : PUTASYNCALLOWED

Skrócona nazwa narzędzia administracyjnego JMS : PAALD

Dostęp programowy

Procedury ustawiające/procedury pobierające

MQDestination.setPutAsyncAllowed()

MQDestination.getPutAsyncAllowed()

Wartości

DEST APS
Określ, czy asynchroniczne operacje umieszczania są dozwolone, odwołując się do definicji kolejki lub tematu. Jest to wartość domyślna.

APS_Q_DEF
Określ, czy asynchroniczne operacje umieszczania są dozwolone, odwołując się do definicji kolejki.

DEFINICJA_TEMATU_AS_TOPIC_DEF
Określ, czy dozwolone są asynchroniczne operacje umieszczania, odwołując się do definicji tematu.

NO
Asynchroniczne operacje umieszczania nie są dozwolone.

YES
Asynchroniczne operacje umieszczania są dozwolone.

QMANAGER

Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie.

Jeśli jednak aplikacja używa tabeli definicji kanału klienta do nawiązania połączenia z menedżerem kolejek, należy zapoznać się z sekcją Używanie tabeli definicji kanału klienta z produktem IBM MQ classes for JMS.

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, TopicConnectionFactory, Queue, XAConnectionFactory, XAQueueConnection Factory, XATopicConnectionFactory

Długa nazwa narzędzia administracyjnego JMS : QMANAGER

Skrócona nazwa narzędzia administracyjnego JMS : QMGR

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setQueueManager ()
- MQConnectionFactory.getQueueManager ()

Wartości

" (pusty łańcuch)
Wartością domyślną może być dowolny łańcuch.

QUEUE

Nazwa miejsca docelowego kolejki produktu JMS . Odpowiada to nazwie kolejki używanej przez menedżer kolejek.

Odpowiednie obiekty

Kolejka

Długa nazwa narzędzia administracyjnego JMS : QUEUE

Skrócona nazwa narzędzia administracyjnego JMS : QU

Wartości

Dowolny łańcuch
Dowolna poprawna nazwa kolejki produktu IBM MQ .

Odsyłacze pokrewne
Reguły nazewnictwa obiektów IBM MQ >

READAHEADALLOWED

Ta właściwość określa, czy konsumenty komunikatów i przeglądarki kolejek mogą używać odczytu z wyprzedzeniem do pobierania nietrwałych komunikatów z tego miejsca docelowego do buforu wewnętrznego przed ich odebraniem.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : READAHEADALLOWED

Skrócona nazwa narzędzia administracyjnego JMS : RAALD

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setReadAheadAllowed()
- MQDestination.getReadAheadAllowed()

Wartości

DEST APS
Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki lub tematu. Jest to wartość domyślna w narzędziach administracyjnych.

W programach należy używać właściwości WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_DEST .

APS_Q_DEF
Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki.

W programach należy używać właściwości WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF .

DEFINICJA_TEMATU_AS_TOPIC_DEF
Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji tematu.

W programach należy używać interfejsu WMQConstants.WMQ_READ_AHEAD_ALLOWED_AS_TOPIC_DEF .

NO
Odczyt z wyprzedzeniem jest niedozwolony.

W programach należy używać właściwości WMQConstants.WMQ_READ_AHEAD_ALLOWED_DISABLED .

YES
Odczyt z wyprzedzeniem jest dozwolony.

W programach należy używać właściwości WMQConstants.WMQ_READ_AHEAD_ALLOWED_ENABLED .

READAHEADCLOSEPOLICY

W przypadku komunikatów dostarczanych do asynchronicznego obiektu nasłuchiwania komunikatów, co dzieje się z komunikatami w wewnętrznym buforze odczytu z wyprzedzeniem, gdy konsument komunikatów jest zamknięty.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : READAHEADCLOSEPOLICY

Skrócona nazwa narzędzia administracyjnego JMS : RACP

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setReadAheadClosePolicy()
- MQDestination.getReadAheadClosePolicy()

Wartości

WSZYSTKO_OGRANICZNIKI
Wszystkie komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem są dostarczane do obiektu nasłuchiwania komunikatów aplikacji przed zwróceniem. Jest to wartość domyślna w narzędziach administracyjnych.

W programach należy używać wartości WMQConstants.WMQ_READ_AHEAD_DELIVERALL .

BIEŻĄCY_SEPARATOR
Tylko bieżące wywołanie obiektu nasłuchiwania komunikatów kończy się przed zwróceniem, potencjalnie pozostawiając komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem, które są następnie usuwane.

W programach należy używać wartości WMQConstants.WMQ_READ_AHEAD_DELIVERCURRENT .

RECEIVECCSID

Właściwość miejsca docelowego, która ustawia docelowy identyfikator CCSID dla konwersji komunikatów menedżera kolejek. Wartość jest ignorowana, chyba że wartość RECEIVECONVERSION jest ustawiona na `WMQ_RECEIVE_CONVERSION_QMGR`

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : RECEIVECCSID

Skrócona nazwa narzędzia administracyjnego JMS : RCCS

Dostęp programowy

Procedury ustawiające/pobierające

- `MQDestination.setReceiveCCSID`
- `MQDestination.getReceiveCCSID`

Wartości

`WMQConstants.WMQ_RECEIVE_CCSID_JVM_DEFAULT`
0 -użyj maszyny JVM `Charset.defaultCharset`

1208
UTF-8

CCSID
Obsługiwany identyfikator kodowanego zestawu znaków.

RECEIVECONVERSION

Właściwość miejsca docelowego, która określa, czy konwersja danych będzie wykonywana przez menedżer kolejek.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : RECEIVECONVERSION

Skrócona nazwa narzędzia administracyjnego JMS : RCNV

Dostęp programowy

Procedury ustawiające/pobierające

- `MQDestination.setReceiveConversion`
- `MQDestination.getReceiveConversion`

Wartości

`WMQConstants.WMQ_RECEIVE_CONVERSION_CLIENT_MSG`
1 -konwersja danych tylko na kliencie z systemem JMS . Wartość domyślna z zakresu od V7.0do 7.0.1.5włącznie.

`WMQConstants.WMQ_RECEIVE_CONVERSION_QMGR`
2 -przed wysłaniem komunikatu do klienta należy wykonać konwersję danych w menedżerze kolejek. Wartość domyślna (i tylko) z wersji V7.0 do wersji V7.0.1.4 włącznie, z wyjątkiem sytuacji, gdy zastosowano poprawkę APAR IC72897 .

RECEIVEISOLATION

Ta właściwość określa, czy subskrybent może odbierać komunikaty, które nie zostały zatwierdzone w kolejce subskrybenta.

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : RECEIVEISOLATION

Skrócona nazwa narzędzia administracyjnego JMS : RCVISOL

Wartości

ZATWIERDZONA
Subskrybent otrzymuje tylko te komunikaty w kolejce subskrybenta, które zostały zatwierdzone. Jest to wartość domyślna w narzędziach administracyjnych.

W programach należy używać wartości WMQConstants.WMQ_RCVISOL_COMMITTED .

NIEZATWIERDZONA
Subskrybent może odbierać komunikaty, które nie zostały zatwierdzone w kolejce subskrybenta.

W programach należy używać właściwości WMQConstants.WMQ_RCVISOL_UNCOMMITTED .

RECEXIT

Identyfikuje wyjście odbierania kanału lub sekwencję wyjść odbierania, które mają być uruchamiane po sobie.

Aby program IBM MQ classes for JMS mógł znaleźć wyjścia odbierania, może być wymagana dodatkowa konfiguracja. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klas IBM MQ dla usługi JMS do używania wyjść kanałów.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : RECEXIT

Skrócona nazwa narzędzia administracyjnego JMS : RCX

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setReceiveExit ()
- MQConnectionFactory.getReceiveExit ()

Wartości

- `null`. Jest to wartość domyślna.
- Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, przy czym każdy element ma jedną z następujących wartości:
  - Nazwa klasy implementującej interfejs `WMQReceiveExit` (dla wyjścia odbierania kanału napisanego w języku Java).
  - Łańcuch w formacie *libraryName(entryPoint)* (dla wyjścia odbierania kanału nie zapisanego w pliku Java).

RECEXITINIT

Dane użytkownika, które są przekazywane do wyjść odbierania kanału podczas ich wywoływania.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : RECEXITINIT

Skrócona nazwa narzędzia administracyjnego JMS : RCXI

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setReceiveExitInit()
- MQConnectionFactory.getReceiveExitInit()

Wartości

null
Łańcuch składający się z jednego lub większej liczby elementów danych użytkownika oddzielonych przecinkami. Jest to wartość domyślna.

REPLYTOSTYLE

Określa sposób konstruowania pola JMSReplyTo w odebranym komunikacie.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : REPLYTOSTYLE

Skrócona nazwa narzędzia administracyjnego JMS : RTOST

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setReplyToStyle()
- MQConnectionFactory.getReplyToStyle()

Wartości

DOMYŚLNA
Odpowiednik `MQMD`.

RFH2
Użyj wartości podanej w nagłówku RFH2 . Jeśli w aplikacji wysyłającej ustawiono wartość JMSReplyTo , należy użyć tej wartości.

MQMD
Użyj wartości podanej w deskryptorze MQMD. To zachowanie jest równoważne domyślnemu zachowaniu produktów IBM WebSphere MQ 6.0.2.4 i 6.0.2.5.

Jeśli wartość JMSReplyTo ustawiona przez aplikację wysyłającą nie zawiera nazwy menedżera kolejek, odbierający menedżer kolejek wstawia własną nazwę do deskryptora MQMD. Jeśli ten parametr zostanie ustawiony na wartość `MQMD`, używana kolejka odpowiedzi znajduje się w odbierającym menedżerze kolejek. Jeśli dla tego parametru zostanie ustawiona wartość RFH2, używana kolejka odpowiedzi znajduje się w menedżerze kolejek określonym w nagłówku RFH2 wysłanego komunikatu, zgodnie z oryginalnie ustawionym przez aplikację wysyłającą.

Jeśli wartość JMSReplyTo ustawiona przez aplikację wysyłającą zawiera nazwę menedżera kolejek, wartość tego parametru jest nieistotne, ponieważ zarówno MQMD, jak i RFH2 zawierają tę samą wartość.

RESCANINT

Jeśli konsument komunikatów w domenie typu punkt z punktem używa selektora komunikatów do wybierania komunikatów, które mają być odbierane, IBM MQ classes for JMS wyszukuje odpowiednie komunikaty w kolejce IBM MQ w kolejności określonej przez atrybut `MsgDeliverySequence` kolejki.

Po IBM MQ classes for JMS znalezieniu odpowiedniego komunikatu i dostarczeniu go do konsumenta IBM MQ classes for JMS wznawia wyszukiwanie następnego odpowiedniego komunikatu od jego bieżącej pozycji w kolejce. Produkt IBM MQ classes for JMS kontynuuje wyszukiwanie w kolejce w ten sposób, dopóki nie osiągnie końca kolejki lub dopóki nie upłynie odstęp czasu w milisekundach określony przez wartość tej właściwości. W każdym przypadku IBM MQ classes for JMS wraca do początku kolejki, aby kontynuować wyszukiwanie i rozpoczyna się nowy przedział czasu.

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Długa nazwa narzędzia administracyjnego JMS : RESCANINT

Skrócona nazwa narzędzia administracyjnego JMS : RINT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setRescanInterwał ()
- MQConnectionFactory.getRescanInterwał ()

Wartości

5000
Wartością domyślną może być dowolna dodatnia liczba całkowita.

SECEXIT

Identyfikuje wyjście zabezpieczeń kanału.

Aby program IBM MQ classes for JMS mógł zlokalizować wyjścia zabezpieczeń, może być wymagana dodatkowa konfiguracja. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klas IBM MQ dla usługi JMS do używania wyjść kanałów.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SECEXIT

Skrócona nazwa narzędzia administracyjnego JMS : SXC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSecurityExit ()
- MQConnectionFactory.getSecurityExit ()

Wartości

- `null`. Jest to wartość domyślna.
- Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, przy czym każdy element ma jedną z następujących wartości:
  - Nazwa klasy implementującej interfejs `WMQSecurityExit` (dla wyjścia zabezpieczeń kanału napisanego w języku Java).
  - Łańcuch w formacie *libraryName(entryPoint)* (dla wyjścia zabezpieczeń kanału nie zapisanego w pliku Java).

SECEXITINIT

Dane użytkownika przekazywane do wyjścia zabezpieczeń kanału w momencie jego wywołania.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SECEXITINIT

Skrócona nazwa narzędzia administracyjnego JMS : SCXI

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSecurityExitInit()
- MQConnectionFactory.getSecurityExitInit()

Wartości

null
Wartością domyślną może być dowolny łańcuch.

SENDCHECKCOUNT

Liczba wywołań wysyłania dozwolonych między sprawdzeniami pod kątem błędów asynchronicznego umieszczania w ramach pojedynczej sesji JMS bez transakcji.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SENDCHECKCOUNT

Skrócona nazwa narzędzia administracyjnego JMS : SCC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSendCheckCount()
- MQConnectionFactory.getSendCheckCount()

Wartości

null
Wartością domyślną może być dowolny łańcuch.

SENDEXIT

Identyfikuje wyjście wysyłania kanału lub sekwencję wyjść wysyłania, które mają być uruchamiane po sobie.

Aby program IBM MQ classes for JMS mógł znaleźć wyjścia wysyłania, może być wymagana dodatkowa konfiguracja. Więcej informacji na ten temat zawiera sekcja Konfigurowanie klas IBM MQ dla usługi JMS do używania wyjść kanałów.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SENDEXIT

Skrócona nazwa narzędzia administracyjnego JMS : SDX

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSendExit ()
- MQConnectionFactory.getSendExit ()

Wartości

- `null`. Jest to wartość domyślna.
- Łańcuch składający się z jednego lub większej liczby elementów oddzielonych przecinkami, przy czym każdy element ma jedną z następujących wartości:
  - Nazwa klasy implementującej interfejs `WMQSendExit` (dla wyjścia wysyłania kanału napisanego w języku Java).
  - Łańcuch w formacie *libraryName(entryPoint)* (dla wyjścia wysyłania kanału nie zapisanego w pliku Java).

SENDEXITINIT

Dane użytkownika przekazywane do wyjść wysyłania kanału w momencie ich wywołania.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SENDEXITINIT

Skrócona nazwa narzędzia administracyjnego JMS : SDXI

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSendExitInit()
- MQConnectionFactory.getSendExitInit()

Wartości

null
Wartością domyślną może być dowolny łańcuch składający się z jednego lub większej liczby elementów danych użytkownika oddzielonych przecinkami.

SHARECONVALLOWED

W przypadku aplikacji, które korzystają z trybu normalnego lub trybu normalnego dostawcy usługi przesyłania komunikatów produktu IBM MQ z ograniczeniami, ta właściwość określa, czy funkcja współużytkowania konwersacji jest używana dla połączeń, sesji i kontekstów produktu JMS utworzonych na podstawie fabryki połączeń.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SHARECONVALLOWED

Skrócona nazwa narzędzia administracyjnego JMS : SCALD

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setShareConvAllowed()
- MQConnectionFactory.getShareConvAllowed()

Wartości

YES

Połączenia, sesje i konteksty produktu JMS utworzone na podstawie fabryki połączeń w ramach tej samej maszyny JVM mogą współużytkować instancję kanału (która jest odwzorowywana na połączenie TCP/IP) w odpowiednich miejscach.

Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQConstants.WMQ_SHARE_CONV_ALLOWED_YES.

NO

Każde połączenie produktu JMS utworzone z fabryki połączeń oraz każda sesja produktu JMS utworzona z tych połączeń produktu JMSma własną instancję kanału (połączenie TCP/IP) z menedżerem kolejek.

W przypadku kontekstów JMS pierwszy kontekst utworzony na podstawie fabryki połączeń tworzy dwie instancje kanału (połączenia TCP/IP). Inne konteksty JMS utworzone na podstawie pierwszego kontekstu mają własną instancję kanału (połączenie TCP/IP).

W przypadku programów należy użyć wartości WMQConstants.WMQ_SHARE_CONV_ALLOWED_NO.

Pojęcia pokrewne

Tryby działania dostawcy przesyłania komunikatów produktu IBM MQ

Współużytkowanie połączenia TCP/IP w klasach IBM MQ classes for JMS

SPARSESUBS

Steruje strategią pobierania komunikatów obiektu TopicSubscriber .

Odpowiednie obiekty

ConnectionFactory, fabryka TopicConnection

Długa nazwa narzędzia administracyjnego JMS : SPARSESUBS

Skrócona nazwa narzędzia administracyjnego JMS : SSUBS

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSparseSubskrypcje ()
- MQConnectionFactory.getSparseSubscriptions ()

Wartości

NO
Subskrypcje odbierają często zgodne komunikaty. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości false.

YES
Subskrypcje otrzymują rzadko zgodne komunikaty. Ta wartość wymaga, aby kolejka subskrypcji mogła zostać otwarta do przeglądania.

W przypadku programów należy użyć wartości true.

SSLCIPHERSUITE

CipherSuite , który ma być używany dla połączenia TLS.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SSLCIPHERSUITE

Skrócona nazwa narzędzia administracyjnego JMS : SCPHS

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSSLCipherSuite ()
- MQConnectionFactory.getSSLCipherSuite ()

Wartości

null
Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja Właściwości TLS obiektów JMS.

SSLCRL

Serwery CRL, które mają zostać sprawdzone pod kątem odwołań certyfikatów TLS.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SSLCRL

Skrócona nazwa narzędzia administracyjnego JMS : SCRL

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSSLCertprzechowuje ()
- MQConnectionFactory.getSSLCertprzechowuje ()

Wartości

null
Rozdzielana spacjami lista adresów URL LDAP. Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja Właściwości TLS obiektów JMS.

SSLFIPSREQUIRED

Ta właściwość określa, czy połączenie TLS musi używać pakietu CipherSuite obsługiwanego przez dostawcę JSSE FIPS (IBMJSSEFIPS) produktu IBM Java .

Uwaga: W systemie AIX, Linux, and Windows IBM MQ zapewnia zgodność ze standardem FIPS 140-2 za pośrednictwem modułu szyfrującego IBM Crypto for C (ICC) . Certyfikat dla tego modułu został przeniesiony do statusu historycznego. Klienci powinni zapoznać się z informacjami w sekcji Certyfikat IBM Crypto for C (ICC) i zapoznać się z poradami NIST. Zastępczy moduł FIPS 140-3 jest obecnie w toku, a jego status można wyświetlić, wyszukując go na liście Moduły NIST CMVP na liście procesów.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SSLFIPSREQUIRED

Skrócona nazwa narzędzia administracyjnego JMS : SFIPS

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSSLFipsWymagana ()
- MQConnectionFactory.getSSLFipsWymagana ()

Wartości

NO
Połączenie TLS może korzystać z dowolnego CipherSuite , który nie jest obsługiwany przez dostawcę JSSE FIPS (IBMJSSEFIPS) w produkcie IBM Java .

Jest to wartość domyślna. W programach należy użyć wartości false.

YES
Połączenie TLS musi używać pakietu CipherSuite obsługiwanego przez serwer IBMJSSEFIPS.

W programach należy użyć wartości true.

SSLPEERNAME

W przypadku protokołu TLS jest to szkielet *nazwy wyróżniającej* , który musi być zgodny z nazwą udostępnioną przez menedżer kolejek.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SSLPEERNAME

Skrócona nazwa narzędzia administracyjnego JMS : SPEER

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSSLPeerNazwa ()
- MQConnectionFactory.getSSLPeerName ()

Wartości

null
Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja Właściwości TLS obiektów JMS.

SSLRESETCOUNT

W przypadku protokołu TLS łączna liczba bajtów wysłanych i odebranych przez połączenie przed ponownym negocjowaniem klucza tajnego używanego do szyfrowania.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SSLRESETCOUNT

Skrócona nazwa narzędzia administracyjnego JMS : SRC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSSLResetLiczebność ()
- MQConnectionFactory.getSSLResetLiczebność ()

Wartości

0
Zero lub dowolna dodatnia liczba całkowita mniejsza lub równa 999, 999, 999. Jest to wartość domyślna. Więcej informacji na ten temat zawiera sekcja Właściwości TLS obiektów JMS.

STATREFRESHINT

Odstęp czasu (w milisekundach) między operacjami odświeżania długotrwałych transakcji, które wykrywają, kiedy subskrybent traci połączenie z menedżerem kolejek.

Ta właściwość ma zastosowanie tylko wtedy, gdy SUBSTORE ma wartość QUEUE.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : STATREFRESHINT

Skrócona nazwa narzędzia administracyjnego JMS : SRI

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setStatusRefreshInterval()
- MQConnectionFactory.getStatusRefreshInterval()

Wartości

60000
Wartością domyślną może być dowolna dodatnia liczba całkowita. Więcej informacji na ten temat zawiera sekcja Właściwości TLS obiektów JMS.

SUBSTORE

Gdzie program IBM MQ classes for JMS przechowuje dane trwałe związane z aktywnymi subskrypcjami.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SUBSTORE

Skrócona nazwa narzędzia administracyjnego JMS : SS

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSubscriptionStore ()
- MQConnectionFactory.getSubscriptionStore ()

Wartości

BROKER
Składnica subskrypcji oparta na brokerze służy do przechowywania szczegółów subskrypcji. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQConstants.WMQ_SUBSTORE_BROKER.

MIGRATE
Prześlij informacje o subskrypcji ze składnicy subskrypcji opartej na kolejce do składnicy subskrypcji opartej na brokerze.

W przypadku programów należy użyć wartości WMQConstants.WMQ_SUBSTORE_MIGRATE.

QUEUE
Składnica subskrypcji oparta na kolejce służy do przechowywania szczegółów subskrypcji.

W przypadku programów należy użyć wartości WMQConstants.WMQ_SUBSTORE_QUEUE.

SYNCPOINTALLGETS

Ta właściwość określa, czy wszystkie pobrania mają być wykonywane w punkcie synchronizacji.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : SYNCPOINTALLGETS

Skrócona nazwa narzędzia administracyjnego JMS : SPAG

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setSyncpointAllGets()
- MQConnectionFactory.getSyncpointAllGets()

Wartości

Nie
Jest to wartość domyślna.

Tak

TARGCLIENT

Ta właściwość określa, czy format IBM MQ RFH2 jest używany do wymiany informacji z aplikacjami docelowymi.

Odpowiednie obiekty

Kolejka, Temat

Długa nazwa narzędzia administracyjnego JMS : TARGCLIENT

Skrócona nazwa narzędzia administracyjnego JMS : TC

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQDestination.setTargetClient()
- MQDestination.getTargetClient()

Wartości

JMS
Obiektem docelowym komunikatu jest aplikacja JMS . Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQConstants.WMQ_CLIENT_JMS_COMPLIANT.

MQ
Obiektem docelowym komunikatu jest aplikacja inna niżJMS IBM MQ .

W przypadku programów należy użyć wartości WMQConstants.WMQ_CLIENT_NONJMS_MQ.

TARGCLIENTMATCHING

Ta właściwość określa, czy komunikat odpowiedzi wysyłany do kolejki identyfikowanej przez pole nagłówka JMSReplyTo komunikatu przychodzącego ma nagłówek MQRFH2 tylko wtedy, gdy komunikat przychodzący ma nagłówek MQRFH2 .

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Długa nazwa narzędzia administracyjnego JMS : TARGCLIENTMATCHING

Skrócona nazwa narzędzia administracyjnego JMS : TCM

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setTargetClientMatching()
- MQConnectionFactory.getTargetClientMatching()

Wartości

YES
Jeśli komunikat przychodzący nie ma nagłówka MQRFH2 , właściwość TARGCLIENT obiektu Queue pochodzącego z pola nagłówka JMSReplyTo komunikatu jest wysyłana do produktu MQ. Jeśli komunikat ma nagłówek MQRFH2 , właściwość TARGCLIENT jest ustawiona na wartość JMS . Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości true.

NO
Właściwość TARGCLIENT obiektu Queue pochodzącego z pola nagłówka JMSReplyTo komunikatu przychodzącego jest zawsze ustawiona na wartość JMS.

W przypadku programów należy użyć wartości false.

TEMPMODEL

Nazwa kolejki modelowej, na podstawie której są tworzone tymczasowe kolejki produktu JMS .

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Długa nazwa narzędzia administracyjnego JMS : TEMPMODEL

Skrócona nazwa narzędzia administracyjnego JMS : TM

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setTemporaryModel ()
- MQConnectionFactory.getTemporaryModel ()

Wartości

SYSTEM.DEFAULT.MODEL.QUEUE
Wartością domyślną może być dowolny łańcuch.

TEMPQPREFIX

Przedrostek używany do tworzenia nazwy kolejki dynamicznej IBM MQ .

Odpowiednie obiekty

ConnectionFactory, QueueConnectionFactory, XAConnectionFactory, XAQueueConnectionFactory

Długa nazwa narzędzia administracyjnego JMS : TEMPQPREFIX

Skrócona nazwa narzędzia administracyjnego JMS : TQP

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setTempQPrefix ()
- MQConnectionFactory.getTempQPrefix ()

Wartości

'' '' (pusty łańcuch)
Używany jest przedrostek `CSQ.*` w systemie z/OS i `AMQ.*` na wszystkich innych platformach. Są to wartości domyślne.

Przedrostek kolejki
Przedrostek kolejki to dowolny łańcuch zgodny z regułami tworzenia treści pola *DynamicQName* w deskryptorze obiektu IBM MQ (struktura MQOD), ale ostatni niepusty znak musi być znakiem gwiazdki.

TEMPTOPICPREFIX

Podczas tworzenia tematów tymczasowych program JMS generuje łańcuch tematu w postaci '' TEMP */TEMPTOPICPREFIX/unique_id* '' lub, jeśli ta właściwość ma wartość domyślną, tylko '' TEMP */unique_id* ''. Określenie niepustego parametru TEMPTOPICPREFIX umożliwia definiowanie konkretnych kolejek modelowych w celu tworzenia kolejek zarządzanych dla subskrybentów tematów tymczasowych utworzonych w ramach tego połączenia.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : TEMPTOPICPREFIX

Skrócona nazwa narzędzia administracyjnego JMS : TTP

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setTempTopicPrefix()
- MQConnectionFactory.getTempTopicPrefix()

Wartości

Dowolny niepusty łańcuch składający się tylko z poprawnych znaków dla łańcucha tematu IBM MQ . Wartością domyślną jest '' '' (pusty łańcuch).

TOPIC

Nazwa miejsca docelowego tematu produktu JMS . Ta wartość jest używana przez menedżer kolejek jako łańcuch tematu publikacji lub subskrypcji.

Odpowiednie obiekty

Temat

Długa nazwa narzędzia administracyjnego JMS : TOPIC

Skrócona nazwa narzędzia administracyjnego JMS : TOP

Wartości

Dowolny łańcuch
Łańcuch, który tworzy poprawny łańcuch tematu IBM MQ . W przypadku używania produktu IBM MQ jako dostawcy przesyłania komunikatów z produktem WebSphere Application Servernależy określić wartość zgodną z nazwą, pod którą temat jest rozpoznawany w celach administracyjnych w produkcie WebSphere Application Server.

Odsyłacze pokrewne
Łańcuchy tematów

TRANSPORT

Rodzaj połączenia z menedżerem kolejek lub brokerem.

Odpowiednie obiekty

ConnectionFactory, Fabryka QueueConnection, Fabryka TopicConnection, XAConnectionFactory, Fabryka XAQueueConnection, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : TRANSPORT

Skrócona nazwa narzędzia administracyjnego JMS : TRAN

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setTransportType ()
- MQConnectionFactory.getTransportType ()

Wartości

BIND
Dla połączenia z menedżerem kolejek w trybie powiązań. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQConstants.WMQ_CM_BINDINGS.

KLIENT
Dla połączenia z menedżerem kolejek w trybie klienta.

W przypadku programów należy użyć wartości WMQConstants.WMQ_CM_CLIENT.

Bezpośrednie
W przypadku połączenia w czasie rzeczywistym z brokerem, który nie używa tunelowania HTTP .

W przypadku programów należy użyć wartości WMQConstants.WMQ_CM_DIRECT_TCPIP.

Usługa DIRECTHTTP
W przypadku połączenia w czasie rzeczywistym z brokerem przy użyciu tunelowania HTTP . Obsługiwany jest tylko protokół HTTP 1.0 .

W przypadku programów należy użyć wartości WMQConstants.WMQ_CM_DIRECT_HTTP.

Pojęcia pokrewne
“Zależności między właściwościami obiektów IBM MQ classes for JMS” na stronie 1956
Poprawność niektórych właściwości zależy od konkretnych wartości innych właściwości.

WILDCARDFORMAT

Ta właściwość określa, która wersja składni znaku wieloznacznego ma być używana.

Odpowiednie obiekty

ConnectionFactory, Fabryka TopicConnection, XAConnectionFactory, Fabryka XATopicConnection

Długa nazwa narzędzia administracyjnego JMS : WILDCARDFORMAT

Skrócona nazwa narzędzia administracyjnego JMS : WCFMT

Dostęp programowy

Procedury ustawiające/procedury pobierające

- MQConnectionFactory.setWildCardFormat()
- MQConnectionFactory.getWildCardFormat()

Wartości

TYLKO_DO_OPR
Rozpoznaje tylko znaki wieloznaczne na poziomie tematu, używane w brokerze w wersji 2. Jest to wartość domyślna dla narzędzi administracyjnych.

W przypadku programów należy użyć wartości WMQConstants.WMQ_WILDCARD_TOPIC_ONLY.

TYLKO_WYKRES
Rozpoznaje tylko znaki wieloznaczne używane w brokerze w wersji 1.

W przypadku programów należy użyć wartości WMQConstants.WMQ_WILDCARD_CHAR_ONLY.

Właściwość ENCODING

Właściwość ENCODING składa się z trzech właściwości podrzędnych, w dwunastu możliwych kombinacjach.

Poprawne wartości, które może przyjąć właściwość ENCODING , są tworzone na podstawie trzech podwłaściwości:

Kodowanie na podstawie liczb całkowitych
Normalny lub odwrócony

Kodowanie dziesiętne
Normalny lub odwrócony

kodowanie zmiennopozycyjne
Normalny IEEE, odwrócony IEEE lub z/OS

Właściwość ENCODING jest wyrażona w postaci łańcucha trzyznakowego z następującą składnią:

```
{N|R}{N|R}{N|R|3}
```

W tym łańcuchu:

- N oznacza normalne
- R oznacza odwrócone
- 3 oznacza z/OS
- Pierwszy znak reprezentuje *kodowanie całkowite*
- Drugi znak reprezentuje *kodowanie dziesiętne*
- Trzeci znak reprezentuje *kodowanie zmiennopozycyjne*

W ten sposób udostępniono zestaw dwunastu możliwych wartości właściwości ENCODING .

Istnieje dodatkowa wartość, łańcuch NATIVE, który ustawia odpowiednie wartości kodowania dla platformy Java .

W poniższych przykładach przedstawiono poprawne kombinacje parametrów ENCODING:

```
ENCODING(NNR)
ENCODING(NATIVE)
ENCODING(RR3)
```

Właściwości TLS obiektów JMS

Włącz szyfrowanie TLS (Transport Layer Security) przy użyciu właściwości SSLCIPHERSUITE. Następnie można zmienić parametry szyfrowania TLS przy użyciu kilku innych właściwości.

Po podaniu wartości TRANSPORT (CLIENT) można włączyć szyfrowaną komunikację TLS za pomocą właściwości SSLCIPHERSUITE. Tę właściwość należy ustawić na poprawną wartość CipherSuite udostępnianą przez dostawcę JSSE. Musi ona być zgodna z nazwą CipherSpec w kanale SVRCONN określoną przez właściwość CHANNEL.

Jednak CipherSpecs (określone w kanale SVRCONN) i CipherSuites (określone w obiektach ConnectionFactory) używają różnych schematów nazewnictwa do reprezentowania tych samych algorytmów szyfrowania TLS. Jeśli rozpoznana nazwa CipherSpec jest określona we właściwości SSLCIPHERSUITE, JMSAdmin wysyła ostrzeżenie i odwzorowuje CipherSpec na odpowiadający jej CipherSuite. Listę CipherSpecs rozpoznawanych przez IBM MQ i JMSAdmin można znaleźć w sekcji TLS CipherSpecs i CipherSuites w IBM MQ classes for JMS .

Jeśli wymagane jest połączenie z użyciem pakietu CipherSuite obsługiwanego przez dostawcę JSSE FIPS IBM Java (IBMJSSEFIPS), należy ustawić właściwość SSLFIPSREQUIRED fabryki połączeń na wartość YES. Wartością domyślną tej właściwości jest NO, co oznacza, że połączenie może korzystać z dowolnego obsługiwanego pakietu CipherSuite. Ta właściwość jest ignorowana, jeśli nie ustawiono właściwości SSLCIPHERSUITE.

Parametr SSLPEERNAME jest zgodny z formatem parametru SSLPEER, który można ustawić w definicjach kanałów. Jest to lista par nazwa-wartość atrybutu oddzielonych przecinkami lub średnikami. Na przykład:

```
SSLPEERNAME(CN=QMGR.*, OU=IBM, OU=WEBSPHERE)
```

Zestaw nazw i wartości tworzy *nazwę wyróżniającą*. Więcej informacji na temat nazw wyróżniających i ich użycia z produktem IBM MQzawiera sekcja Zabezpieczanie produktu IBM MQ.

Podany przykład sprawdza certyfikat identyfikujący prezentowany przez serwer w czasie połączenia. Aby połączenie powiodło się, certyfikat musi mieć nazwę zwykłą rozpoczynającą się od QMGR., i musi mieć co najmniej dwie nazwy jednostek organizacyjnych, z których pierwsza to IBM , a druga to WEBSPHERE. Podczas sprawdzania nie jest rozróżniana wielkość liter.

Jeśli parametr SSLPEERNAME nie jest ustawiony, takie sprawdzanie nie jest wykonywane. Parametr SSLPEERNAME jest ignorowany, jeśli parametr SSLCIPHERSUITE nie jest ustawiony.

Właściwość SSLCRL określa zero lub więcej serwerów CRL (Certificate Revocation List). Użycie tej właściwości wymaga użycia maszyny JVM w wersji Java 2 v1.4. Jest to rozdzielana spacjami lista wpisów w postaci:

```
ldap:// hostname:[ port ]
```

opcjonalnie po którym następuje pojedynczy znak/. Jeśli parametr *port* zostanie pominięty, przyjmowany jest domyślny port LDAP 389. W czasie połączenia certyfikat TLS prezentowany przez serwer jest porównywany z określonymi serwerami CRL. Więcej informacji na temat zabezpieczeń CRL zawiera sekcja Zabezpieczanie produktu IBM MQ .

Jeśli wartość SSLCRL nie jest ustawiona, takie sprawdzanie nie jest wykonywane. Opcja SSLCRL jest ignorowana, jeśli opcja SSLCIPHERSUITE nie jest ustawiona.

Właściwość SSLRESETCOUNT reprezentuje łączną liczbę bajtów wysłanych i odebranych przez połączenie przed ponownym negocjowaniem klucza tajnego używanego do szyfrowania. Liczba wysłanych bajtów jest liczbą przed szyfrowaniem, a liczba odebranych bajtów jest liczbą po deszyfrowaniu. Liczba bajtów obejmuje również informacje sterujące wysyłane i odbierane przez IBM MQ classes for JMS.

Na przykład, aby skonfigurować obiekt ConnectionFactory , który może być używany do tworzenia połączenia przez kanał MQI z włączoną obsługą TLS z kluczem tajnym, który jest renegocjowany po przekazaniu 4 MB danych, należy wydać następującą komendę w narzędziu JMSAdmin:

```
ALTER CF(my.cf) SSLRESETCOUNT(4194304)
```

Jeśli wartość parametru SSLRESETCOUNT wynosi zero, co jest wartością domyślną, klucz tajny nigdy nie jest renegocjowany. Właściwość SSLRESETCOUNT jest ignorowana, jeśli nie ustawiono właściwości SSLCIPHERSUITE.

Informacje uzupełniające dotyczące produktu IBM MQ Message Service Client (XMS) for .NET

Ta sekcja zawiera informacje na temat interfejsów klas IBM MQ Message Service Client (XMS) for .NET (XMS .NET) i właściwości obiektów zdefiniowanych przez XMS.

.NET interfejsy

W tej sekcji opisano interfejsy klas .NET oraz ich właściwości i metody.

Poniższa tabela zawiera podsumowanie interfejsów zdefiniowanych w przestrzeni nazw `IBM.XMS` .

Tabela 871. Podsumowanie interfejsów klasy .NET

Interfejs	**Opis**
"IBytesMessage" na stronie 2013	Komunikat bajtowy to komunikat, którego treść składa się ze strumienia bajtów.
"IPołączenie" na stronie 2023	Obiekt połączenia reprezentuje aktywne połączenie aplikacji z serwerem przesyłania komunikatów.
"IConnectionFactory" na stronie 2026	Aplikacja używa fabryki połączeń do utworzenia połączenia.
"Dane interfejsu IConnectionMeta" na stronie 2027	Obiekt danych ConnectionMetaudostępnia informacje o połączeniu.
"IMiejsce docelowe" na stronie 2028	Miejsce docelowe to miejsce, z którego aplikacja wysyła komunikaty, lub miejsce źródłowe, z którego aplikacja odbiera komunikaty, lub oba te miejsca.
"ExceptionListener" na stronie 2029	Aplikacja używa obiektu nasłuchiwania wyjątków do asynchronicznego powiadamiania o problemie z połączeniem.
"Wyjątek IllegalState" na stronie 2029	XMS zgłasza ten wyjątek, jeśli aplikacja wywołuje metodę w niepoprawnym lub nieodpowiednim czasie lub jeśli XMS nie jest w stanie odpowiednim dla żądanej operacji.
"InitialContext" na stronie 2030	Aplikacja używa obiektu InitialContext do tworzenia obiektów na podstawie definicji obiektów pobieranych z repozytorium obiektów administrowanych.
"Wyjątek IDException InvalidClient" na stronie 2032	Produkt XMS zgłasza ten wyjątek, jeśli aplikacja próbuje ustawić identyfikator klienta dla połączenia, ale identyfikator klienta jest niepoprawny lub jest już używany.
"Wyjątek InvalidDestination" na stronie 2032	Produkt XMS zgłasza ten wyjątek, jeśli aplikacja określa niepoprawne miejsce docelowe.
"Wyjątek InvalidSelector" na stronie 2032	Program XMS zgłasza ten wyjątek, jeśli aplikacja udostępnia wyrażenie selektora komunikatów, którego składnia jest niepoprawna.
"IMapMessage" na stronie 2032	Komunikat odwzorowania to komunikat, którego treść składa się z zestawu par nazwa-wartość, z którym każda wartość ma powiązany typ danych.

Tabela 871. Podsumowanie interfejsów klasy .NET (kontynuacja)	
Interfejs	**Opis**
“IKomunikat” na stronie 2041	Obiekt Message reprezentuje komunikat wysyłany lub odbierany przez aplikację. IMessage jest nadklasą dla klas komunikatów, takich jak IMapMessage.
“IMessageConsumer” na stronie 2047	Aplikacja używa konsumenta komunikatów do odbierania komunikatów wysyłanych do miejsca docelowego.
“MessageEOFException” na stronie 2050	Program XMS zgłasza ten wyjątek, jeśli podczas odczytywania przez aplikację treści komunikatu bajtowego program XMS napotka koniec strumienia komunikatu bajtowego.
“Wyjątek MessageFormat” na stronie 2050	Program XMS zgłasza ten wyjątek, jeśli program XMS napotka komunikat w niepoprawnym formacie.
“IMessageListener (delegowanie)” na stronie 2050	Aplikacja używa obiektu nasłuchiwania komunikatów do asynchronicznego odbierania komunikatów.
“MessageNotReadableException” na stronie 2051	Produkt XMS zgłasza ten wyjątek, jeśli aplikacja próbuje odczytać treść komunikatu, który jest tylko do zapisu.
“MessageNotWritableException” na stronie 2051	Produkt XMS zgłasza ten wyjątek, jeśli aplikacja próbuje dokonać zapisu w treści komunikatu, który jest tylko do odczytu.
“IMessageProducer” na stronie 2051	Aplikacja używa producenta komunikatów do wysyłania komunikatów do miejsca docelowego.
“IObjectMessage” na stronie 2057	Komunikat obiektu to komunikat, którego treść składa się z obiektu Java lub .NET przekształconego do postaci szeregowej.
“IPropertyContext” na stronie 2058	IPropertyContext jest abstrakcyjną nadklasą zawierającą metody służące do pobierania i ustawiania właściwości. Metody te są dziedziczone przez inne klasy.
“IQueueBrowser” na stronie 2066	Aplikacja używa przeglądarki kolejek do przeglądania komunikatów w kolejce bez ich usuwania.
“Żądający” na stronie 2068	Aplikacja używa requestera do wysyłania komunikatu żądania, a następnie oczekiwania na odpowiedź i jej odbierania.
“Wyjątek ResourceAllocation” na stronie 2069	Program XMS zgłasza ten wyjątek, jeśli program XMS nie może przydzielić zasobów wymaganych przez metodę.

Tabela 871. Podsumowanie interfejsów klasy .NET (kontynuacja)	
Interfejs	**Opis**
“SecurityException” na stronie 2070	XMS zgłasza ten wyjątek, jeśli identyfikator użytkownika i hasło podane w celu uwierzytelnienia aplikacji zostaną odrzucone. XMS zgłasza również ten wyjątek, jeśli sprawdzenie uprawnień nie powiedzie się i metoda nie zostanie zakończona.
“ISesja” na stronie 2070	Sesja jest jednowątkowym kontekstem do wysyłania i odbierania komunikatów.
“IStreamMessage” na stronie 2080	Komunikat strumienia to komunikat, którego treść składa się ze strumienia wartości, z którym każda wartość ma powiązany typ danych.
“ITextMessage” na stronie 2089	Komunikat tekstowy to komunikat, którego treść składa się z łańcucha.
“TransactionInProgressException” na stronie 2090	Program XMS zgłasza ten wyjątek, jeśli aplikacja żąda operacji, która nie jest poprawna, ponieważ transakcja jest w toku.
“TransactionRolledBackException” na stronie 2090	XMS zgłasza ten wyjątek, jeśli aplikacja wywołuje funkcję Session.commit() w celu zatwierdzenia bieżącej transakcji, ale transakcja jest wycofywana.
XMSC	W przypadku systemu .NETnazwy i wartości właściwości XMS są definiowane w tej klasie jako stałe publiczne. Więcej informacji na ten temat zawiera sekcja “Właściwości obiektów XMS” na stronie 2093.
“Wyjątek XMSException” na stronie 2090	Jeśli program XMS wykryje błąd podczas przetwarzania wywołania metody .NET , program XMS zgłosi wyjątek. Wyjątkiem jest obiekt, który hermetyzuje informacje o błędzie. Istnieją różne typy wyjątków XMS , a obiekt XMSException jest tylko jednym z typów wyjątków. Jednak klasa XMSException jest nadklasą innych klas wyjątków XMS . XMS zgłasza wyjątek XMSException w sytuacjach, w których żaden inny typ wyjątku nie jest odpowiedni.
“XMSFactoryFactory” na stronie 2091	Jeśli aplikacja nie używa obiektów administrowanych, należy użyć tej klasy do tworzenia fabryk połączeń, kolejek i tematów.

Definicja każdej metody zawiera listę kodów wyjątków, które XMS może zwrócić, jeśli wykryje błąd podczas przetwarzania wywołania metody. Każdy kod wyjątku jest reprezentowany przez stałą nazwaną, która ma odpowiedni wyjątek.

IBytesMessage

Komunikat bajtowy to komunikat, którego treść składa się ze strumienia bajtów.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
```

```
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IBytesMessage
```

.NET właściwości

BodyLength -Pobierz długość treści

Interfejs:

```
Int64 BodyLength
    {
      get;
    }
```

Pobierz długość treści komunikatu (w bajtach), gdy treść komunikatu jest tylko do odczytu.

Zwracana wartość jest długością całej treści, niezależnie od tego, gdzie aktualnie znajduje się kursor do odczytu komunikatu.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException

metody

ReadBoolean -wartość boolowska odczytu

Interfejs:

```
Boolean ReadBoolean();
```

Odczytaj wartość boolowską ze strumienia komunikatów bajtów.

Parametry:
Brak

Zwraca:
Wartość boolowska, która jest odczytywana.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadSignedByte-Odczyt Byte

Interfejs:

```
Int16   ReadSignedByte();
```

Odczytaj następny bajt ze strumienia komunikatów bajtów jako 8-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytany bajt.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes -Odczytane Bajty

Interfejs:

```
Int32   ReadBytes(Byte[] array);
Int32   ReadBytes(Byte[] array, Int32 length);
```

Odczytaj tablicę bajtów ze strumienia komunikatów bajtowych, rozpoczynając od bieżącej pozycji kursora.

Parametry:

array (wyjście)
Bufor, który ma zawierać tablicę odczytanych bajtów. Jeśli liczba bajtów pozostałych do odczytania ze strumienia przed wywołaniem jest większa lub równa długości buforu, bufor jest wypełniany. W przeciwnym razie bufor zostanie częściowo zapełniony wszystkimi pozostałymi bajtami.

Jeśli na wejściu zostanie określony wskaźnik pusty, metoda pominie bajty bez ich odczytu. Jeśli liczba bajtów pozostałych do odczytania ze strumienia przed wywołaniem jest większa lub równa długości buforu, liczba pominiętych bajtów jest równa długości buforu. W przeciwnym razie wszystkie pozostałe bajty zostaną pominięte. Kursor pozostaje na następnej pozycji do odczytania w strumieniu komunikatów bajtowych.

długość (wejście)
Długość buforu w bajtach

Zwraca:
Liczba bajtów, które zostały wczytane do buforu. Jeśli bufor jest częściowo zapełniony, wartość jest mniejsza niż długość buforu, co oznacza, że nie ma więcej bajtów do odczytania. Jeśli przed wywołaniem nie ma już żadnych bajtów do odczytania ze strumienia, wartością jest XMSC_END_OF_STREAM.

Jeśli na wejściu zostanie określony wskaźnik pusty, metoda nie zwróci żadnej wartości.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException

ReadChar -Odczyt znaku

Interfejs:

```
Char    ReadChar();
```

Odczytaj następne 2 bajty ze strumienia komunikatów bajtów jako znak.

Parametry:
Brak

Zwraca:
Odczytany znak.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble -odczyt liczby zmiennopozycyjnej o podwójnej precyzji

Interfejs:

```
Double  ReadDouble();
```

Odczytaj następne 8 bajtów ze strumienia komunikatów bajtów jako liczbę zmiennopozycyjną podwójnej precyzji.

Parametry:
Brak

Zwraca:
Liczba zmiennopozycyjna o podwójnej precyzji, która jest odczytywana.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat -odczyt liczby zmiennopozycyjnej

Interfejs:

```
Single  ReadFloat();
```

Odczytaj następne 4 bajty ze strumienia komunikatów jako liczbę zmiennopozycyjną.

Parametry:
Brak

Zwraca:
Liczba zmiennopozycyjna, która jest odczytywana.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt -Odczyt liczby całkowitej

Interfejs:

```
Int32   ReadInt();
```

Odczytaj następne 4 bajty ze strumienia komunikatów jako 32-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytana liczba całkowita.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong -Odczyt Long Integer

Interfejs:

```
Int64   ReadLong();
```

Odczytaj następne 8 bajtów ze strumienia komunikatów bajtowych jako 64-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytana liczba całkowita typu long.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadShort -Odczyt Short Integer (krótka liczba całkowita)

Interfejs:

```
Int16   ReadShort();
```

Odczytaj następne 2 bajty ze strumienia komunikatów jako 16-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytana krótka liczba całkowita.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte -Odczyt bajtu bez znaku

Interfejs:

```
Byte    ReadByte();
```

Odczyt następnego bajtu ze strumienia komunikatów bajtów jako 8-bitowej liczby całkowitej bez znaku.

Parametry:
Brak

Zwraca:
Odczytany bajt.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadUnsignedShort-Read Unsigned Short Integer (krótka liczba całkowita bez znaku)

Interfejs:

```
Int32   ReadUnsignedShort();
```

Odczytaj następne 2 bajty ze strumienia komunikatów jako 16-bitową liczbę całkowitą bez znaku.

Parametry:
Brak

Zwraca:
Odczytana krótka liczba całkowita bez znaku.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadUTF -odczyt łańcucha UTF

Interfejs:

```
String  ReadUTF();
```

Odczytaj łańcuch, zakodowany w UTF-8, ze strumienia komunikatów bajtów.

Uwaga: Przed wywołaniem metody ReadUTF() należy upewnić się, że kursor buforu wskazuje na początek strumienia komunikatów bajtowych.

Parametry:
Brak

Zwraca:
Obiekt typu String obudowujący odczytany łańcuch.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

Resetuj-Resetuj

Interfejs:

```
void Reset();
```

Przełącz treść komunikatu w tryb tylko do odczytu i zmień pozycję kursora na początku strumienia komunikatów bajtów.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException

WriteBoolean -wartość boolowska zapisu

Interfejs:

```
void  WriteBoolean(Boolean value);
```

Zapisz wartość boolowską w strumieniu komunikatów bajtów.

Parametry:

wartość (dane wejściowe)
Wartość boolowska, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteByte -Zapis w bajtach

Interfejs:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Zapisz bajt w strumieniu komunikatów bajtów.

Parametry:

wartość (dane wejściowe)
Bajt, który ma zostać zapisany.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteBytes -Zapisane bajty (Write Bytes)

Interfejs:

```
void  WriteBytes(Byte[] value);
```

Zapisz tablicę bajtów w strumieniu komunikatów bajtów.

Parametry:

wartość (dane wejściowe)
Tablica bajtów do zapisania.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteBytes -zapis częściowej tablicy bajtów

Interfejs:

```
void  WriteBytes(Byte[] value, int offset, int length);
```

Zapisz część tablicy bajtów w bajtowym strumieniu komunikatów, zgodnie z definicją o określonej długości.

Parametry:

wartość (dane wejściowe)
Tablica bajtów do zapisania.

przesunięcie (wejście)
Punkt początkowy dla tablicy bajtów, które mają zostać zapisane.

długość (wejście)
Liczba bajtów do zapisu.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteChar -Zapisz znak

Interfejs:

```
void  WriteChar(Char value);
```

Zapisz znak w strumieniu komunikatów bajtów jako 2 bajty, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Znak, który ma zostać zapisany.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteDouble -liczba zmiennopozycyjna o podwójnej precyzji dla zapisu

Interfejs:

```
void  WriteDouble(Double value);
```

Przekształć liczbę zmiennopozycyjną o podwójnej precyzji w długą liczbę całkowitą i zapisz długą liczbę całkowitą w bajtowym strumieniu komunikatów jako 8 bajtów, najpierw w bajcie o najwyższej kolejności.

Parametry:

wartość (dane wejściowe)
Liczba zmiennopozycyjna podwójnej precyzji, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteFloat -liczba zmiennopozycyjna zapisu

Interfejs:

```
void  WriteFloat(Single value);
```

Przekształć liczbę zmiennopozycyjną w liczbę całkowitą i zapisz liczbę całkowitą w bajtowym strumieniu komunikatów jako 4 bajty, najpierw bajt najstarszy.

Parametry:

wartość (dane wejściowe)
Liczba zmiennopozycyjna, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteInt -Write Integer (liczba całkowita)

Interfejs:

```
void  WriteInt(Int32 value);
```

Zapisz liczbę całkowitą w bajtowym strumieniu komunikatów jako 4 bajty, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Liczba całkowita do zapisania.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteLong -Zapis Long Integer

Interfejs:

```
void  WriteLong(Int64 value);
```

Zapisz długą liczbę całkowitą w bajtowym strumieniu komunikatów jako 8 bajtów, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Liczba całkowita typu long, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteObject -zapis obiektu

Interfejs:

```
void  WriteObject(Object value);
```

Zapisz określony obiekt w strumieniu komunikatu bajtowego.

Parametry:

wartość (dane wejściowe)
Obiekt, który ma zostać zapisany, który musi być odwołaniem do typu podstawowego.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteShort -Short Write Integer (krótka liczba całkowita)

Interfejs:

```
void  WriteShort(Int16 value);
```

W strumieniu komunikatu bajtowego należy zapisać krótką liczbę całkowitą jako 2 bajty, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Krótka liczba całkowita, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteUTF -Zapisz łańcuch UTF

Interfejs:

```
void  WriteUTF(String value);
```

Zapisz łańcuch, zakodowany w UTF-8, do strumienia komunikatów bajtów.

Parametry:

wartość (dane wejściowe)
Obiekt typu String obudowujący łańcuch, który ma zostać zapisany.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

- MessageNotWritableException

Dziedziczone właściwości i metody

Następujące właściwości zostały odziedziczone po interfejsie IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimdanym, JMSType, Właściwości

Następujące metody zostały odziedziczone po interfejsie IMessage:

clearBody, clearProperties, PropertyExists

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPołączenie

Obiekt połączenia reprezentuje aktywne połączenie aplikacji z serwerem przesyłania komunikatów.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnection
```

Listę zdefiniowanych przez XMS właściwości obiektu połączenia zawiera sekcja “Właściwości połączenia” na stronie 2094.

.NET właściwości

ClientID -pobieranie i ustawianie identyfikatora klienta

Interfejs:

```
String ClientID
      {
        get;
        set;
      }
```

Pobierz i ustaw identyfikator klienta dla połączenia.

Identyfikator klienta może być wstępnie skonfigurowany przez administratora w fabryce ConnectionFactorylub przypisany przez ustawienie ClientID.

Identyfikator klienta jest używany tylko do obsługi trwałych subskrypcji w domenie publikowania/subskrypcji i jest ignorowany w domenie typu punkt z punktem.

Jeśli aplikacja ustawia identyfikator klienta dla połączenia, musi to zrobić natychmiast po utworzeniu połączenia i przed wykonaniem jakiejkolwiek innej operacji na połączeniu. Jeśli aplikacja podejmie próbę ustawienia identyfikatora klienta po tym punkcie, wywołanie zgłosi wyjątek IllegalState.

Ta właściwość nie jest poprawna w przypadku połączenia z brokerem w czasie rzeczywistym.

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState
- Wyjątek IDException InvalidClient

ExceptionListener -pobieranie i ustawianie obiektu nasłuchiwania wyjątków

Interfejs:

```
ExceptionListener ExceptionListener
    {
      get;
      set;
    }
```

Pobierz obiekt nasłuchiwania wyjątków, który jest zarejestrowany w połączeniu, i zarejestruj obiekt nasłuchiwania wyjątków w połączeniu.

Jeśli w połączeniu nie jest zarejestrowany żaden obiekt nasłuchiwania wyjątków, metoda zwraca wartość NULL. Jeśli obiekt nasłuchiwania wyjątków jest już zarejestrowany w połączeniu, można anulować rejestrację, podając wartość NULL zamiast obiektu nasłuchiwania wyjątków.

Więcej informacji na temat używania obiektów nasłuchiwania wyjątków zawiera sekcja Używanie obiektów nasłuchiwania wyjątków i komunikatów w produkcie .NET.

Wyjątki:

- Wyjątek XMSException

Metadane-pobieranie metadanych

Interfejs:

```
IConnectionMetaData MetaData
    {
      get;
    }
```

Pobierz metadane dla połączenia.

Wyjątki:

- Wyjątek XMSException

metody

Zamknij-Zamknij połączenie

Interfejs:

```
void  Close();
```

Zamknij połączenie.

Jeśli aplikacja próbuje zamknąć połączenie, które jest już zamknięte, wywołanie jest ignorowane.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

CreateSession -Tworzenie sesji

Interfejs:

```
ISession CreateSession(Boolean transacted,
                       AcknowledgeMode acknowledgeMode);
```

Utwórz sesję.

Parametry:

transakcyjne (wejście)
Wartość `True` oznacza, że sesja jest transakcją. Wartość `False` oznacza, że sesja nie jest transakcją.

W przypadku połączenia w czasie rzeczywistym z brokerem wartością musi być `False`.

acknowledgeMode (wejście)
Wskazuje, w jaki sposób potwierdzane są komunikaty odbierane przez aplikację. Wartość musi być jedną z następujących wartości z wyliczenia AcknowledgeMode :

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

W przypadku połączenia w czasie rzeczywistym z brokerem wartością musi być `AcknowledgeMode.AutoAcknowledge` lub `AcknowledgeMode.DupsOkAcknowledge`

Ten parametr jest ignorowany, jeśli sesja jest transakcją. Więcej informacji na temat trybów potwierdzania zawiera sekcja Potwierdzanie komunikatów.

Zwraca:
Obiekt sesji

Wyjątki:

- Wyjątek XMSException

Uruchom-Uruchom połączenie

Interfejs:

```
void  Start();
```

Uruchom lub zrestartuj dostarczanie komunikatów przychodzących dla połączenia. Wywołanie jest ignorowane, jeśli połączenie jest już uruchomione.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

Zatrzymaj-zatrzymaj połączenie

Interfejs:

```
void  Stop();
```

Zatrzymaj dostarczanie komunikatów przychodzących dla połączenia. Wywołanie jest ignorowane, jeśli połączenie zostało już zatrzymane.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IConnectionFactory

Aplikacja używa fabryki połączeń do utworzenia połączenia.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionFactory
```

Listę XMS zdefiniowanych właściwości obiektu ConnectionFactory zawiera sekcja “Właściwości obiektu ConnectionFactory” na stronie 2094.

metody

CreateConnection -Utwórz fabrykę połączeń (przy użyciu domyślnej tożsamości użytkownika)

Interfejs:

```
IConnection CreateConnection();
```

Utwórz fabrykę połączeń z właściwościami domyślnymi.

Jeśli nawiązywane jest połączenie z programem IBM MQ , a parametr XMSC_USERID nie jest ustawiony, menedżer kolejek domyślnie używa parametru userID zalogowanego użytkownika. Jeśli wymagane jest dalsze uwierzytelnianie pojedynczych użytkowników na poziomie połączenia, można napisać wyjście uwierzytelniania klienta skonfigurowane w programie IBM MQ.

Parametry:
Brak

Wyjątki:

- Wyjątek XMSException

CreateConnection -Utwórz połączenie (przy użyciu określonej tożsamości użytkownika)

Interfejs:

```
IConnection CreateConnection(String userId, String password);
```

Utwórz połączenie przy użyciu określonej tożsamości użytkownika.

Jeśli nawiązywane jest połączenie z programem IBM MQ , a parametr XMSC_USERID nie jest ustawiony, menedżer kolejek domyślnie używa parametru userID zalogowanego użytkownika. Jeśli wymagane jest dalsze uwierzytelnianie pojedynczych użytkowników na poziomie połączenia, można napisać wyjście uwierzytelniania klienta skonfigurowane w programie IBM MQ.

Połączenie jest tworzone w trybie zatrzymania. Żadne komunikaty nie będą dostarczane, dopóki aplikacja nie wywoła wywołania **`Connection.start()`**.

Parametry:

userID (wejście)
Obiekt typu String obudowujący identyfikator użytkownika, który ma być używany do uwierzytelniania aplikacji. W przypadku podania wartości NULL podejmowana jest próba utworzenia połączenia bez uwierzytelniania.

hasło (dane wejściowe)
Obiekt typu String obudowujący hasło, które ma być używane do uwierzytelniania aplikacji. W przypadku podania wartości NULL podejmowana jest próba utworzenia połączenia bez uwierzytelniania.

Zwraca:
Obiekt połączenia.

Wyjątki:

- Wyjątek XMSException
- XMS_X_SECURITY_EXCEPTION

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Dane interfejsu IConnectionMeta

Obiekt danych ConnectionMetaudostępnia informacje o połączeniu.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IConnectionMetaData
```

Lista XMS zdefiniowanych właściwości obiektu danych ConnectionMetaznajduje się w sekcji “Właściwości danych ConnectionMeta” na stronie 2099.

.NET właściwości

JMSXPropertyNames -Pobieranie właściwości komunikatu zdefiniowanego przez JMS

Interfejs:

```
System.Collections.IEnumerator JMSXPropertyNames
    {
      get;
    }
```

Zwraca wyliczenie nazw JMS zdefiniowanych właściwości komunikatu obsługiwanych przez połączenie.

Zdefiniowane właściwości komunikatu JMS nie są obsługiwane przez połączenie z brokerem w czasie rzeczywistym.

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMiejsce docelowe

Miejsce docelowe to miejsce, z którego aplikacja wysyła komunikaty, lub miejsce źródłowe, z którego aplikacja odbiera komunikaty, lub oba te miejsca.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IDestination
```

Listę zdefiniowanych XMS właściwości obiektu docelowego zawiera sekcja “Właściwości miejsca docelowego” na stronie 2100.

.NET właściwości

Nazwa-Pobierz nazwę miejsca docelowego

Interfejs:

```
String Name
{
  get;
}
```

Pobierz nazwę miejsca docelowego. Nazwa jest łańcuchem zawierającym nazwę kolejki lub nazwę tematu.

Wyjątki:

- Wyjątek XMSException

TypeId -pobierz typ miejsca docelowego

Interfejs:

```
DestinationType TypeId
{
  get;
}
```

Pobierz typ miejsca docelowego. Typ miejsca docelowego może przyjmować jedną z następujących wartości:

```
DestinationType.Queue
DestinationType.Topic
```

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ExceptionListener

Aplikacja używa obiektu nasłuchiwania wyjątków do asynchronicznego powiadamiania o problemie z połączeniem.

Hierarchia dziedziczenia:
Brak

Jeśli aplikacja używa połączenia tylko w celu asynchronicznego odbierania komunikatów, a nie do innych celów, jedynym sposobem, w jaki aplikacja może uzyskać informacje o problemie z połączeniem, jest użycie obiektu nasłuchiwania wyjątków. W innych sytuacjach proces nasłuchiwania wyjątków może udostępnić bardziej natychmiastowy sposób uzyskania informacji o problemie z połączeniem niż oczekiwanie na następne wywołanie synchroniczne do XMS.

Delegowanie

ExceptionListener -obiekt nasłuchiwania wyjątków

Interfejs:

```
public delegate void ExceptionListener(Exception ex)
```

Powiadom aplikację o problemie z połączeniem.

Metody implementujące tego delegata mogą być zarejestrowane w połączeniu.

Więcej informacji na temat używania obiektów nasłuchiwania wyjątków zawiera sekcja Używanie obiektów nasłuchiwania wyjątków i komunikatów w produkcie .NET.

Parametry:

wyjątek (dane wejściowe)
Wskaźnik do wyjątku utworzonego przez XMS.

Zwraca:
Unieważnione

Wyjątek IllegalState

XMS zgłasza ten wyjątek, jeśli aplikacja wywołuje metodę w niepoprawnym lub nieodpowiednim czasie lub jeśli XMS nie jest w stanie odpowiednim dla żądanej operacji.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.Exception
           |
           +----IBM.XMS.IllegalStateException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

InitialContext

Aplikacja używa obiektu InitialContext do tworzenia obiektów na podstawie definicji obiektów pobieranych z repozytorium obiektów administrowanych.

Hierarchia dziedziczenia:
Brak

.NET właściwości

Środowisko-Pobierz środowisko

Interfejs:

```
Hashtable Environment
{
     get;
}
```

Pobierz środowisko.

Wyjątki:

- Wyjątki są specyficzne dla używanej usługi katalogowej.

Konstruktory

InitialContext -Tworzy kontekst początkowy

Interfejs:

```
InitialContext(Hashtable env);
```

Utwórz obiekt InitialContext .

Parametry:

Informacje wymagane do nawiązania połączenia z repozytorium obiektów administrowanych są udostępniane konstruktorowi w tabeli mieszającej środowiska.

Wyjątki:

- Wyjątek XMSException

metody

AddToEnvironment-dodawanie nowej właściwości do środowiska

Interfejs:

```
Object AddToEnvironment(String propName, Object propVal);
```

Dodaj nową właściwość do środowiska.

Parametry:

propName (dane wejściowe)
Obiekt typu String zawierający nazwę właściwości, która ma zostać dodana.

propVal (wejście)
Wartość właściwości, która ma zostać dodana.

Zwraca:
Stara wartość właściwości.

Wyjątki:

- Wyjątki są specyficzne dla używanej usługi katalogowej.

Zamknij-zamknij ten kontekst

Interfejs:

```
void Close()
```

Zamknij ten kontekst.

Parametry:
Brak

Zwraca:
Brak

Wyjątki:

- Wyjątki są specyficzne dla używanej usługi katalogowej.

Wyszukiwanie-wyszukiwanie obiektu w kontekście początkowym

Interfejs:

```
Object Lookup(String name);
```

Utwórz obiekt na podstawie definicji obiektu pobranej z repozytorium obiektów administrowanych.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę obiektu administrowanego, który ma zostać pobrany. Nazwa może być nazwą prostą lub nazwą złożoną. Więcej informacji na ten temat zawiera sekcja Pobieranie obiektów administrowanych.

Zwraca:
Obiekt IConnectionFactory lub obiekt IDestination, w zależności od typu pobieranego obiektu. Jeśli funkcja może uzyskać dostęp do katalogu, ale nie może znaleźć wymaganego obiektu, zwracana jest wartość NULL.

Wyjątki:

- Wyjątki są specyficzne dla używanej usługi katalogowej.

RemoveFromEnvironment-usuwanie właściwości ze środowiska

Interfejs:

```
Object RemoveFromEnvironment(String propName);
```

Usuń właściwość ze środowiska.

Parametry:

propName (dane wejściowe)
Obiekt typu String zawierający nazwę właściwości, która ma zostać usunięta.

Zwraca:
Obiekt, który został usunięty.

Wyjątki:

- Wyjątki są specyficzne dla używanej usługi katalogowej.

Wyjątek IDException InvalidClient

Produkt XMS zgłasza ten wyjątek, jeśli aplikacja próbuje ustawić identyfikator klienta dla połączenia, ale identyfikator klienta jest niepoprawny lub jest już używany.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidClientIDException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

Wyjątek InvalidDestination

Produkt XMS zgłasza ten wyjątek, jeśli aplikacja określa niepoprawne miejsce docelowe.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidDestinationException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

Wyjątek InvalidSelector

Program XMS zgłasza ten wyjątek, jeśli aplikacja udostępnia wyrażenie selektora komunikatów, którego składnia jest niepoprawna.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.InvalidSelectorException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

IMapMessage

Komunikat odwzorowania to komunikat, którego treść składa się z zestawu par nazwa-wartość, z którym każda wartość ma powiązany typ danych.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IMapMessage
```

Gdy aplikacja pobiera wartość z pary nazwa-wartość, wartość może zostać przekształcona przez XMS w inny typ danych. Więcej informacji na temat tej formy konwersji niejawnej zawiera sekcja dotycząca komunikatów odwzorowania w temacie Treść komunikatu XMS.

.NET właściwości

MapNames -Pobieranie nazw odwzorowań

Interfejs:

```
System.Collections.IEnumerator MapNames
{
  get;
}
```

Pobierz wyliczenie nazw w treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

metody

GetBoolean -pobierz wartość boolowską

Interfejs:

```
Boolean GetBoolean(String name);
```

Pobierz wartość boolowską identyfikowaną przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą wartość boolowską.

Zwraca:
Wartość boolowska pobrana z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetByte -Pobieranie Bajtu

Interfejs:

```
Byte    GetByte(String name);
    Int16   GetSignedByte(String name);
```

Pobierz bajt identyfikowany przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String obudowujący nazwę, która identyfikuje bajt.

Zwraca:
Bajt pobrany z treści komunikatu odwzorowania. Na bajcie nie jest wykonywana żadna konwersja danych.

Wyjątki:

- Wyjątek XMSException

GetBytes -Pobieranie bajtów

Interfejs:

```
Byte[]  GetBytes(String name);
```

Pobierz tablicę bajtów identyfikowanych przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą tablicę bajtów.

Zwraca:
Liczba bajtów w tablicy.

Wyjątki:

- Wyjątek XMSException

GetChar -Pobranie znaku

Interfejs:

```
Char    GetChar(String name);
```

Pobierz znak identyfikowany przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą znak.

Zwraca:
Znak pobrany z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetDouble -Uzyskanie liczby zmiennopozycyjnej o podwójnej precyzji

Interfejs:

```
Double  GetDouble(String name);
```

Pobierz liczbę zmiennopozycyjną podwójnej precyzji identyfikowaną przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String obudowujący nazwę, która identyfikuje liczbę zmiennopozycyjną podwójnej precyzji.

Zwraca:
Liczba zmiennopozycyjna o podwójnej precyzji pobrana z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetFloat -Uzyskanie liczby zmiennopozycyjnej

Interfejs:

```
Single  GetFloat(String name);
```

Pobierz liczbę zmiennopozycyjną identyfikowaną przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą liczbę zmiennopozycyjną.

Zwraca:
Liczba zmiennopozycyjna pobrana z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetInt -Pobieranie liczby całkowitej

Interfejs:

```
Int32   GetInt(String name);
```

Pobierz liczbę całkowitą identyfikowaną przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą liczbę całkowitą.

Zwraca:
Liczba całkowita pobrana z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetLong -Pobranie Long Integer

Interfejs:

```
Int64   GetLong(String name);
```

Pobierz długą liczbę całkowitą identyfikowaną przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą długą liczbę całkowitą.

Zwraca:
Długa liczba całkowita pobrana z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetObject -Pobranie obiektu

Interfejs:

```
Object  GetObject(String name);
```

Uzyskaj odwołanie do wartości pary nazwa-wartość z treści komunikatu odwzorowania. Para nazwa-wartość jest identyfikowana przez nazwę.

Parametry:

nazwa (wejście)
Obiekt typu String obudowujący nazwę pary nazwa-wartość.

Zwraca:
Wartość, która jest jednym z następujących typów obiektów:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Wyjątki:
Wyjątek XMSException

GetShort -pobranie krótkiej liczby całkowitej

Interfejs:

```
Int16   GetShort(String name);
```

Pobierz krótką liczbę całkowitą identyfikowaną przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String obudowujący nazwę, która identyfikuje krótką liczbę całkowitą.

Zwraca:
Krótka liczba całkowita pobrana z treści komunikatu odwzorowania.

Wyjątki:

- Wyjątek XMSException

GetString -pobranie łańcucha

Interfejs:

```
String  GetString(String name);
```

Pobierz łańcuch identyfikowany przez nazwę z treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą łańcuch w treści komunikatu odwzorowania.

Zwraca:
Obiekt typu String obudowujący łańcuch pobrany z treści komunikatu odwzorowania. Jeśli konwersja danych jest wymagana, ta wartość jest łańcuchem po konwersji.

Wyjątki:

- Wyjątek XMSException

ItemExists -sprawdzanie istnienia pary nazwa-wartość

Interfejs:

```
Boolean ItemExists(String name);
```

Sprawdź, czy treść komunikatu odwzorowania zawiera parę nazwa-wartość o podanej nazwie.

Parametry:

nazwa (wejście)
Obiekt typu String obudowujący nazwę pary nazwa-wartość.

Zwraca:

- `True`, jeśli treść komunikatu odwzorowania zawiera parę nazwa-wartość o podanej nazwie.
- `False`, jeśli treść komunikatu odwzorowania nie zawiera pary nazwa-wartość o podanej nazwie.

Wyjątki:

- Wyjątek XMSException

SetBoolean -ustaw wartość boolowską

Interfejs:

```
void  SetBoolean(String name, Boolean value);
```

Ustaw wartość boolowską w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą wartość boolowską w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Wartość boolowska do ustawienia.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetByte -Ustaw bajt

Interfejs:

```
void  SetByte(String name, Byte value);
void  SetSignedByte(String name, Int16 value);
```

Ustaw bajt w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą bajt w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Bajt, który ma zostać ustawiony.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetBytes -Ustaw bajty

Interfejs:

```
void  SetBytes(String name, Byte[] value);
```

Ustaw tablicę bajtów w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą tablicę bajtów w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Tablica bajtów do ustawienia.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetChar -znak zestawu

Interfejs:

```
void  SetChar(String name, Char value);
```

Ustaw 2-bajtowy znak w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą znak w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Znak, który ma zostać ustawiony.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetDouble -ustawienie liczby zmiennopozycyjnej o podwójnej precyzji

Interfejs:

```
void  SetDouble(String name, Double value);
```

Ustaw liczbę zmiennopozycyjną podwójnej precyzji w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String obudowujący nazwę w celu zidentyfikowania liczby zmiennopozycyjnej o podwójnej precyzji w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Liczba zmiennopozycyjna podwójnej precyzji, która ma zostać ustawiona.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetFloat -ustaw liczbę zmiennopozycyjną

Interfejs:

```
void  SetFloat(String name, Single value);
```

Ustaw liczbę zmiennopozycyjną w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą liczbę zmiennopozycyjną w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Liczba zmiennopozycyjna, która ma zostać ustawiona.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetInt -Ustawianie liczby całkowitej

Interfejs:

```
void  SetInt(String name, Int32 value);
```

Ustaw liczbę całkowitą w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą liczbę całkowitą w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Liczba całkowita do ustawienia.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetLong -Ustaw Long Integer

Interfejs:

```
void  SetLong(String name, Int64 value);
```

Ustaw długą liczbę całkowitą w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą długą liczbę całkowitą w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Liczba całkowita typu long, która ma zostać ustawiona.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetObject -ustawienie obiektu

Interfejs:

```
void  SetObject(String name, Object value);
```

Ustaw wartość, która musi być typem podstawowym XMS w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą wartość w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Tablica bajtów zawierająca wartość, która ma zostać ustawiona.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetShort -ustaw krótką liczbę całkowitą

Interfejs:

```
void  SetShort(String name, Int16 value);
```

Ustaw krótką liczbę całkowitą w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą krótką liczbę całkowitą w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Krótka liczba całkowita do ustawienia.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

SetString -ustawienie łańcucha

Interfejs:

```
void  SetString(String name, String value);
```

Ustaw łańcuch w treści komunikatu odwzorowania.

Parametry:

nazwa (wejście)
Obiekt typu String zawierający nazwę identyfikującą łańcuch w treści komunikatu odwzorowania.

wartość (dane wejściowe)
Obiekt typu String obudowujący łańcuch, który ma zostać ustawiony.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące właściwości zostały odziedziczone po interfejsie IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimdanym, JMSType, Właściwości

Następujące metody zostały odziedziczone po interfejsie IMessage:

clearBody, clearProperties, PropertyExists

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IKomunikat

Obiekt Message reprezentuje komunikat wysyłany lub odbierany przez aplikację. IMessage jest nadklasą dla klas komunikatów, takich jak IMapMessage.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
```

Lista pól nagłówka komunikatu JMS w obiekcie komunikatu znajduje się w sekcji Pola nagłówka komunikatu XMS. Lista JMS zdefiniowanych właściwości obiektu Message znajduje się w sekcji Zdefiniowane przez JMS właściwości komunikatu. Lista IBM zdefiniowanych właściwości obiektu Message znajduje się w sekcji IBM-defined properties of a message(Właściwości zdefiniowane przez IBM). Listę właściwości JMS_IBM_MQMD* obiektu Message zawiera sekcja “Właściwości JMS_IBM_MQMD*” na stronie 2104

Komunikaty są usuwane przez funkcję czyszczenia pamięci. Usunięcie komunikatu powoduje zwolnienie zasobów, których używał.

.NET właściwości

GetJMSCorrelationIdentyfikator-Pobierz i ustaw JMSCorrelationID

Interfejs:

```
String JMSCorrelationID
{
  get;
  set;
}
```

Pobierz i ustaw identyfikator korelacji komunikatu jako obiekt typu String.

Wyjątki:

- Wyjątek XMSException

JMSDeliveryMode -pobieranie i ustawianie właściwości JMSDeliveryMode

Interfejs:

```
DeliveryMode JMSDeliveryMode
{
  get;
  set;
}
```

Pobierz i ustaw tryb dostarczania komunikatu.

Tryb dostarczania komunikatu może przyjmować jedną z następujących wartości:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

Dla nowo utworzonej wiadomości, która nie została wysłana, trybem dostarczania jest `DeliveryMode.Trwała`, z wyjątkiem połączenia w czasie rzeczywistym z brokerem, dla którego trybem dostarczania jest `DeliveryMode.NonPersistent`. Dla odebranego komunikatu metoda zwraca tryb dostarczania ustawiony przez wywołanie IMessageProducer.send () podczas wysyłania komunikatu, chyba że aplikacja odbierająca zmieni tryb dostarczania przez ustawienie JMSDeliveryMode.

Wyjątki:

- Wyjątek XMSException

JMSDestination-pobieranie i ustawianie miejsca JMSDestination

Interfejs:

```
IDestination JMSDestination
{
  get;
  set;
}
```

Pobierz i ustaw miejsce docelowe komunikatu.

Miejsce docelowe jest ustawiane przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu. Wartość JMSDestination jest ignorowana. Można jednak użyć miejsca JMSDestination, aby zmienić miejsce docelowe odebranego komunikatu.

W przypadku nowo utworzonego komunikatu, który nie został wysłany, metoda zwraca obiekt miejsca docelowego o wartości NULL, chyba że aplikacja wysyłająca ustawia miejsce docelowe, ustawiając miejsce docelowe JMSDestination. W przypadku odebranego komunikatu metoda zwraca obiekt miejsca docelowego dla miejsca docelowego, które zostało ustawione przez wywołanie metody

IMessageProducer.send () podczas wysyłania komunikatu, chyba że aplikacja odbierająca zmieni miejsce docelowe, ustawiając miejsce docelowe JMSDestination.

Wyjątki:

- Wyjątek XMSException

JMSExpiration-pobieranie i ustawianie wartości JMSExpiration

Interfejs:

```
Int64 JMSExpiration
{
  get;
  set;
}
```

Pobierz i ustaw czas ważności komunikatu.

Czas utraty ważności jest ustawiany przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu. Jego wartość jest obliczana przez dodanie czasu życia, określonego przez aplikację wysyłającą, do czasu wysłania komunikatu. Czas ważności jest wyrażony w milisekundach od godziny 00:00:00 GMT w dniu 1 stycznia 1970.

Dla nowo utworzonego komunikatu, który nie został wysłany, czas ważności wynosi 0, chyba że aplikacja wysyłająca ustawi inny czas ważności, ustawiając wartość JMSExpiration. Dla odebranego komunikatu metoda zwraca czas ważności, który został ustawiony przez wywołanie IMessageProducer.send () podczas wysyłania komunikatu, chyba że aplikacja odbierająca zmieni czas ważności przez ustawienie JMSExpiration.

Jeśli czas życia wynosi 0, wywołanie metody IMessageProducer.send () ustawia czas ważności na 0, aby wskazać, że komunikat nie traci ważności.

Produkt XMS usuwa komunikaty, które utraciły ważność, i nie dostarcza ich do aplikacji.

Wyjątki:

- Wyjątek XMSException

JMSMessageID -pobieranie i ustawianie JMSMessageID

Interfejs:

```
String JMSMessageID
{
  get;
  set;
}
```

Pobierz i ustaw identyfikator komunikatu jako obiekt łańcuchowy obudowujący identyfikator komunikatu.

Identyfikator komunikatu jest ustawiany przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu. Dla odebranego komunikatu metoda zwraca identyfikator komunikatu, który został ustawiony przez wywołanie IMessageProducer.send () podczas wysyłania komunikatu, chyba że aplikacja odbierająca zmieni identyfikator komunikatu przez ustawienie JMSMessageID.

Jeśli komunikat nie ma identyfikatora, metoda zwraca wartość NULL.

Wyjątki:

- Wyjątek XMSException

JMSPriority-pobieranie i ustawianie atrybutu JMSPriority.

Interfejs:

```
Int32 JMSPriority
```

```
{
  get;
  set;
}
```

Pobierz i ustaw priorytet komunikatu.

Priorytet jest ustawiany przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu. Wartość jest liczbą całkowitą z zakresu od 0(najniższy priorytet) do 9(najwyższy priorytet).

Dla nowo utworzonego komunikatu, który nie został wysłany, priorytet ma wartość 4 , chyba że aplikacja wysyłająca ustawi inny priorytet, ustawiając wartość JMSPriority. W przypadku odebranego komunikatu metoda zwraca priorytet ustawiony przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu, chyba że aplikacja odbierająca zmieni priorytet, ustawiając atrybut JMSPriority.

Wyjątki:

- Wyjątek XMSException

JMSRedelivered-Pobieranie i ustawianie JMSRedelivered

Interfejs:

```
Boolean JMSRedelivered
{
  get;
  set;
}
```

Uzyskaj informację o tym, czy komunikat jest ponownie dostarczany, i wskaż, czy komunikat jest ponownie dostarczany. Wskazanie jest ustawiane przez wywołanie metody IMessageConsumer.receive () po odebraniu komunikatu.

Ta właściwość ma następujące wartości:

- `True`, jeśli komunikat jest ponownie dostarczany.
- `False`, jeśli komunikat nie jest ponownie dostarczany.

W przypadku połączenia w czasie rzeczywistym z brokerem wartością jest zawsze `False`.

Wskazanie ponownego dostarczenia ustawione przez interfejs JMSRedelivered przed wysłaniem komunikatu jest ignorowane przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu oraz jest ignorowane i zastępowane przez wywołanie metody IMessageConsumer.receive () po odebraniu komunikatu. Można jednak użyć klasy JMSRedelivered do zmiany wskazania odebranego komunikatu.

Wyjątki:

- Wyjątek XMSException

JMSReplyTo -pobieranie i ustawianie właściwości JMSReplyTo

Interfejs:

```
IDestination JMSReplyTo
{
  get;
  set;
}
```

Pobierz i ustaw miejsce docelowe, do którego ma zostać wysłana odpowiedź na komunikat.

Wartością tej właściwości jest obiekt miejsca docelowego, do którego ma zostać wysłana odpowiedź na komunikat. Obiekt docelowy o wartości NULL oznacza, że nie jest oczekiwana żadna odpowiedź.

Wyjątki:

- Wyjątek XMSException

JMSTim-właściwości-Pobierz i ustaw JMSTim-właściwości

Interfejs:

```
Int64 JMSTimestamp
{
  get;
  set;
}
```

Pobierz i ustaw czas wysłania komunikatu.

Znacznik czasu jest ustawiany przez wywołanie metody IMessageProducer.send (), gdy komunikat jest wysyłany, i jest wyrażany w milisekundach od godziny 00:00:00 GMT w dniu 1 stycznia 1970.

W przypadku nowo utworzonego komunikatu, który nie został wysłany, znacznik czasu ma wartość 0, chyba że aplikacja wysyłająca ustawi inny znacznik czasu, ustawiając wartość JMSTim/To.
W przypadku odebranego komunikatu metoda zwraca znacznik czasu ustawiony przez wywołanie metody IMessageProducer.send () podczas wysyłania komunikatu, chyba że aplikacja odbierająca zmieni znacznik czasu, ustawiając właściwość JMSTim/To.

Wyjątki:

- Wyjątek XMSException

Uwagi:

1. Jeśli znacznik czasu nie jest zdefiniowany, metoda zwraca wartość 0, ale nie zgłasza wyjątku.

JMSType-Pobierz i ustaw JMSType

Interfejs:

```
String JMSType
{
  get;
  set;
}
```

Pobierz i ustaw typ komunikatu.

Wartość atrybutu JMSType jest łańcuchem obudowującym typ komunikatu. Jeśli konwersja danych jest wymagana, ta wartość jest typem po konwersji.

Wyjątki:

- Wyjątek XMSException

PropertyNames -Pobieranie właściwości

Interfejs:

```
System.Collections.IEnumerator PropertyNames
{
  get;
}
```

Pobieranie wyliczenia właściwości nazw komunikatu.

Wyjątki:

- Wyjątek XMSException

metody

Potwierdzenie-potwierdzenie

Interfejs:

```
void  Acknowledge();
```

Potwierdź ten komunikat i wszystkie wcześniej niepotwierdzone komunikaty odebrane przez sesję.

Aplikacja może wywołać tę metodę, jeśli tryb potwierdzenia sesji to AcknowledgeMode.ClientAcknowledge. Wywołania metody są ignorowane, jeśli sesja ma inny tryb potwierdzenia lub jest transakcją.

Komunikaty, które zostały odebrane, ale nie zostały potwierdzone, mogą zostać ponownie dostarczone.

Więcej informacji na temat potwierdzania komunikatów zawiera sekcja ../develop/xms_cmesack.dita#xms_cmesack.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState

ClearBody -czyszczenie treści

Interfejs:

```
void  ClearBody();
```

Wyczyść treść komunikatu. Pola nagłówka i właściwości komunikatu nie są czyszczone.

Jeśli aplikacja wyczyści treść komunikatu, pozostaje ona w tym samym stanie, co pusta treść w nowo utworzonym komunikacie. Stan pustej treści w nowo utworzonym komunikacie zależy od typu treści komunikatu. Więcej informacji na ten temat zawiera sekcja Treść komunikatu XMS.

Aplikacja może wyczyścić treść komunikatu w dowolnym momencie, bez względu na stan, w jakim znajduje się treść. Jeśli treść komunikatu jest tylko do odczytu, jedynym sposobem, w jaki aplikacja może zapisywać treść, jest wcześniejsze usunięcie treści przez aplikację.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

ClearProperties -Wyczyść właściwości

Interfejs:

```
void  ClearProperties();
```

Wyczyść właściwości komunikatu. Pola nagłówka i treść komunikatu nie są czyszczone.

Jeśli aplikacja wyczyści właściwości komunikatu, właściwości te stają się dostępne do odczytu i zapisu.

Aplikacja może w dowolnym momencie wyczyścić właściwości komunikatu, bez względu na stan, w jakim znajdują się właściwości. Jeśli właściwości komunikatu są tylko do odczytu, jedynym sposobem, w jaki właściwości mogą być dostępne do zapisu, jest wcześniejsze usunięcie właściwości przez aplikację.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

PropertyExists -sprawdzanie istnienia właściwości

Interfejs:

```
Boolean PropertyExists(String propertyName);
```

Sprawdź, czy komunikat ma właściwość o podanej nazwie.

Parametry:

propertyName (dane wejściowe)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:

- `True`, jeśli komunikat ma właściwość o podanej nazwie.
- `False`, jeśli komunikat nie zawiera właściwości o podanej nazwie.

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IMessageConsumer

Aplikacja używa konsumenta komunikatów do odbierania komunikatów wysyłanych do miejsca docelowego.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageConsumer
```

Listę zdefiniowanych XMS właściwości obiektu MessageConsumer zawiera sekcja “Właściwości elementu MessageConsumer” na stronie 2107.

.NET właściwości

MessageListener -pobieranie i ustawianie obiektu nasłuchiwania komunikatów

Interfejs:

```
MessageListener MessageListener
{
  get;
  set;
}
```

Pobierz obiekt nasłuchiwania komunikatów, który jest zarejestrowany w konsumencie komunikatów, i zarejestruj obiekt nasłuchiwania komunikatów w konsumencie komunikatów.

Jeśli w konsumencie komunikatów nie zarejestrowano żadnego obiektu nasłuchiwania komunikatów, parametr MessageListener ma wartość NULL. Jeśli obiekt nasłuchiwania komunikatów jest już zarejestrowany w konsumencie komunikatów, można anulować rejestrację, podając zamiast tego wartość NULL.

Więcej informacji na temat korzystania z obiektów nasłuchiwania komunikatów zawiera sekcja Używanie obiektów nasłuchiwania komunikatów i wyjątków w środowisku .NET.

Wyjątki:

- Wyjątek XMSException

MessageSelector -pobieranie selektora komunikatów

Interfejs:

```
String MessageSelector
{
  get;
}
```

Pobierz selektor komunikatów dla konsumenta komunikatów. Wartością zwracaną jest obiekt typu String obudowujący wyrażenie selektora komunikatów. Jeśli konwersja danych jest wymagana, ta wartość jest wyrażeniem selektora komunikatów po konwersji. Jeśli konsument komunikatów nie ma selektora komunikatów, wartością MessageSelector jest obiekt typu String o wartości NULL.

Wyjątki:

- Wyjątek XMSException

metody

Zamknij-Zamknij konsument komunikatów

Interfejs:

```
void  Close();
```

Zamknij konsument komunikatów.

Jeśli aplikacja próbuje zamknąć konsumenta komunikatów, który jest już zamknięty, wywołanie jest ignorowane.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

Odbierz-Odbierz

Interfejs:

```
IMessage Receive();
```

Odbierz następny komunikat dla konsumenta komunikatów. Wywołanie oczekuje na komunikat w nieskończoność lub do momentu zamknięcia konsumenta komunikatów.

Parametry:
Brak

Zwraca:
Wskaźnik do obiektu Message. Jeśli konsument komunikatów jest zamknięty, gdy wywołanie oczekuje na komunikat, metoda zwraca wskaźnik do obiektu Message o wartości NULL.

Wyjątki:

- Wyjątek XMSException

Odbieranie-odbieranie (z przedziałem czasu oczekiwania)

Interfejs:

```
IMessage Receive(Int64 delay);
```

Odbierz następny komunikat dla konsumenta komunikatów. Wywołanie oczekuje tylko przez określony czas na komunikat lub do momentu zamknięcia konsumenta komunikatów.

Parametry:

opóźnienie (wejście)
Czas w milisekundach, przez który wywołanie oczekuje na komunikat. Jeśli zostanie podany przedział czasu oczekiwania 0, wywołanie będzie oczekiwać na komunikat w nieskończoność.

Zwraca:
Wskaźnik do obiektu Message. Jeśli w okresie oczekiwania nie zostanie odebrany żaden komunikat lub jeśli konsument komunikatów jest zamknięty w czasie, gdy wywołanie oczekuje na komunikat, metoda zwraca wskaźnik do obiektu komunikatu o wartości NULL, ale nie zgłasza wyjątku.

Wyjątki:

- Wyjątek XMSException

ReceiveNoWait-Receive No Wait (Brak oczekiwania)

Interfejs:

```
IMessage ReceiveNoWait();
```

Odbierz następny komunikat dla konsumenta komunikatów, jeśli jest on natychmiast dostępny.

Parametry:
Brak

Zwraca:
Wskaźnik do obiektu komunikatu. Jeśli komunikat nie jest natychmiast dostępny, metoda zwraca wskaźnik do obiektu komunikatu o wartości NULL.

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

MessageEOFException

Program XMS zgłasza ten wyjątek, jeśli podczas odczytywania przez aplikację treści komunikatu bajtowego program XMS napotka koniec strumienia komunikatu bajtowego.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageEOFException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

Wyjątek MessageFormat

Program XMS zgłasza ten wyjątek, jeśli program XMS napotka komunikat w niepoprawnym formacie.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageFormatException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

IMessageListener (delegowanie)

Aplikacja używa obiektu nasłuchiwania komunikatów do asynchronicznego odbierania komunikatów.

Hierarchia dziedziczenia:
Brak

Delegowanie

MessageListener -obiekt nasłuchiwania komunikatów

Interfejs:

```
public delegate void MessageListener(IMessage msg);
```

Dostarcz komunikat asynchronicznie do konsumenta komunikatów.

Metody implementujące tego delegata mogą być zarejestrowane w połączeniu.

Więcej informacji na temat używania obiektów nasłuchiwania komunikatów zawiera sekcja Używanie obiektów nasłuchiwania komunikatów i wyjątków w produkcie .NET.

Parametry:

mesg (wejście)
Obiekt Message.

Zwraca:
Unieważnione

MessageNotReadableException

Produkt XMS zgłasza ten wyjątek, jeśli aplikacja próbuje odczytać treść komunikatu, który jest tylko do zapisu.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotReadableException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

MessageNotWritableException

Produkt XMS zgłasza ten wyjątek, jeśli aplikacja próbuje dokonać zapisu w treści komunikatu, który jest tylko do odczytu.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.MessageNotWritableException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

IMessageProducer

Aplikacja używa producenta komunikatów do wysyłania komunikatów do miejsca docelowego.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessageProducer
```

Listę zdefiniowanych XMS właściwości obiektu MessageProducer zawiera sekcja “Właściwości producenta MessageProducer” na stronie 2108.

.NET właściwości

DeliveryMode -pobieranie i ustawianie domyślnego trybu dostarczania

Interfejs:

```
DeliveryMode DeliveryMode
{
  get;
  set;
}
```

Pobierz i ustaw domyślny tryb dostarczania dla komunikatów wysyłanych przez producenta komunikatów.

Domyślny tryb dostarczania ma jedną z następujących wartości:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

W przypadku połączenia w czasie rzeczywistym z brokerem wartością musi być `DeliveryMode.NonPersistent`.

Wartością domyślną jest `DeliveryMode.Persistent`, z wyjątkiem połączenia w czasie rzeczywistym z brokerem, dla którego wartością domyślną jest `DeliveryMode.NonPersistent`.

Wyjątki:

- Wyjątek XMSException

Miejsce docelowe-Pobierz miejsce docelowe

Interfejs:

```
IDestination Destination
{
  get;
}
```

Pobierz miejsce docelowe dla producenta komunikatów.

Parametry:
Brak

Zwraca:
Obiekt docelowy. Jeśli producent komunikatów nie ma miejsca docelowego, metoda zwraca obiekt miejsca docelowego o wartości NULL.

Wyjątki:

- Wyjątek XMSException

DisableMsgID-pobranie i ustawienie flagi wyłączenia ID komunikatu

Interfejs:

```
Boolean DisableMessageID
{
  get;
  set;
}
```

Uzyskaj wskazanie, czy aplikacja odbierająca wymaga, aby identyfikatory komunikatów były dołączane do komunikatów wysyłanych przez producenta komunikatów, oraz wskazanie, czy aplikacja odbierająca wymaga, aby identyfikatory komunikatów były dołączane do komunikatów wysyłanych przez producenta komunikatów.

W przypadku połączenia z menedżerem kolejek lub w czasie rzeczywistym połączenia z brokerem ta flaga jest ignorowana. W przypadku połączenia z magistralą integracji usług flaga jest uwzględniana.

Identyfikator DisabledMsgma następujące wartości:

- `True`, jeśli aplikacja odbierająca nie wymaga, aby identyfikatory komunikatów były dołączane do komunikatów wysyłanych przez producenta komunikatów.
- `False`, jeśli aplikacja odbierająca wymaga, aby identyfikatory komunikatów były uwzględniane w komunikatach wysyłanych przez producenta komunikatów.

Wyjątki:

- Wyjątek XMSException

DisableMsgTS-pobranie i ustawienie flagi wyłączenia znacznika czasu

Interfejs:

```
Boolean DisableMessageTimestamp
{
  get;
  set;
}
```

Pobierz wskazanie, czy aplikacja odbierająca wymaga, aby znaczniki czasu były uwzględniane w komunikatach wysyłanych przez producenta komunikatów, oraz wskazanie, czy aplikacja odbierająca wymaga, aby znaczniki czasu były uwzględniane w komunikatach wysyłanych przez producenta komunikatów.

W przypadku połączenia w czasie rzeczywistym z brokerem ta flaga jest ignorowana. Flaga jest uwzględniana w przypadku połączenia z menedżerem kolejek lub połączenia z magistralą integracji usług.

Parametr DisableMsgTS ma następujące wartości:

- `True`, jeśli aplikacja odbierająca nie wymaga, aby znaczniki czasu były uwzględniane w komunikatach wysyłanych przez producenta komunikatów.
- `False`, jeśli aplikacja odbierająca wymaga, aby znaczniki czasu były uwzględniane w komunikatach wysyłanych przez producenta komunikatów.

Zwraca:

Wyjątki:

- Wyjątek XMSException

Priorytet-pobieranie i ustawianie priorytetu domyślnego

Interfejs:

```
Int32 Priority
{
  get;
  set;
}
```

Pobierz i ustaw domyślny priorytet komunikatów wysyłanych przez producenta komunikatów.

Wartość domyślnego priorytetu komunikatu jest liczbą całkowitą z zakresu od `0`(najniższy priorytet) do 9(najwyższy priorytet).

W przypadku połączenia w czasie rzeczywistym z brokerem priorytet komunikatu jest ignorowany.

Wyjątki:

- Wyjątek XMSException

TimeTo-pobieranie i ustawianie domyślnego czasu życia

Interfejs:

```
Int64 TimeToLive
{
```

```
  get;
  set;
}
```

Pobierz i ustaw domyślny czas istnienia komunikatu, zanim utraci on ważność.

Czas jest mierzony od momentu wysłania komunikatu przez producenta komunikatów i jest to domyślny czas życia (w milisekundach). Wartość 0 oznacza, że komunikat nigdy nie traci ważności.

W przypadku połączenia w czasie rzeczywistym z brokerem ta wartość zawsze wynosi 0.

Wyjątki:

- Wyjątek XMSException

metody

Zamknij-zamknij producenta komunikatów

Interfejs:

```
void  Close();
```

Zamknij producenta komunikatów.

Jeśli aplikacja próbuje zamknąć producenta komunikatów, który jest już zamknięty, wywołanie jest ignorowane.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

Wyślij-Wyślij

Interfejs:

```
void Send(IMessage msg) ;
```

Wyślij komunikat do miejsca docelowego określonego podczas tworzenia producenta komunikatów. Wyślij komunikat przy użyciu domyślnego trybu dostarczania producenta komunikatów, priorytetu i czasu życia.

Parametry:

komunikat (wejście)
Obiekt Message.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek MessageFormat
- Wyjątek InvalidDestination

Wysyłanie-wysyłanie (określenie trybu dostarczania, priorytetu i czasu życia)

Interfejs:

```
void Send(IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive);
```

Wyślij komunikat do miejsca docelowego określonego podczas tworzenia producenta komunikatów. Wyślij komunikat przy użyciu określonego trybu dostarczania, priorytetu i czasu życia.

Parametry:

komunikat (wejście)
Obiekt Message.

deliveryMode (wejście)
Tryb dostarczania komunikatu, który musi mieć jedną z następujących wartości:

`DeliveryMode.Persistent`
`DeliveryMode.NonPersistent`

W przypadku połączenia w czasie rzeczywistym z brokerem wartością musi być `DeliveryMode.NonPersistent`.

priorytet (wejście)
Priorytet komunikatu. Wartość może być liczbą całkowitą z zakresu od 0dla najniższego priorytetu do 9dla najwyższego priorytetu. W przypadku połączenia z brokerem w czasie rzeczywistym wartość ta jest ignorowana.

timeToLive (wejście)
Czas życia komunikatu w milisekundach. Wartość 0 oznacza, że komunikat nigdy nie traci ważności. W przypadku połączenia w czasie rzeczywistym z brokerem wartość musi wynosić 0.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek MessageFormat
- Wyjątek InvalidDestination
- Wyjątek IllegalState

Wyślij-Wyślij (do określonego miejsca docelowego)

Interfejs:

```
void Send(IDestination dest, IMessage msg) ;
```

Wyślij komunikat do określonego miejsca docelowego, jeśli używany jest producent komunikatów, dla którego podczas tworzenia producenta komunikatów nie określono miejsca docelowego. Wyślij komunikat przy użyciu domyślnego trybu dostarczania producenta komunikatów, priorytetu i czasu życia.

Zwykle miejsce docelowe jest określane podczas tworzenia producenta komunikatów, ale w przeciwnym razie należy określić miejsce docelowe za każdym razem, gdy wysyłany jest komunikat.

Parametry:

Dest (wejście)
Obiekt docelowy.

komunikat (wejście)
Obiekt Message.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek MessageFormat
- Wyjątek InvalidDestination

Wyślij-wyślij (do określonego miejsca docelowego, określając tryb dostarczania, priorytet i czas życia)

Interfejs:

```
void Send(IDestination dest,
          IMessage msg,
          DeliveryMode deliveryMode,
          Int32 priority,
          Int64 timeToLive) ;
```

Wyślij komunikat do określonego miejsca docelowego, jeśli używany jest producent komunikatów, dla którego podczas tworzenia producenta komunikatów nie określono miejsca docelowego. Wyślij komunikat przy użyciu określonego trybu dostarczania, priorytetu i czasu życia.

Zwykle miejsce docelowe jest określane podczas tworzenia producenta komunikatów, ale w przeciwnym razie należy określić miejsce docelowe za każdym razem, gdy wysyłany jest komunikat.

Parametry:

Dest (wejście)
Obiekt docelowy.

komunikat (wejście)
Obiekt Message.

deliveryMode (wejście)
Tryb dostarczania komunikatu, który musi mieć jedną z następujących wartości:

```
DeliveryMode.Persistent
DeliveryMode.NonPersistent
```

W przypadku połączenia w czasie rzeczywistym z brokerem wartością musi być `DeliveryMode.NonPersistent`.

priorytet (wejście)
Priorytet komunikatu. Wartość może być liczbą całkowitą z zakresu od 0dla najniższego priorytetu do 9dla najwyższego priorytetu. W przypadku połączenia z brokerem w czasie rzeczywistym wartość ta jest ignorowana.

timeToLive (wejście)
Czas życia komunikatu w milisekundach. Wartość 0 oznacza, że komunikat nigdy nie traci ważności. W przypadku połączenia w czasie rzeczywistym z brokerem wartość musi wynosić 0.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek MessageFormat
- Wyjątek InvalidDestination
- Wyjątek IllegalState

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IObjectMessage

Komunikat obiektu to komunikat, którego treść składa się z obiektu Java lub .NET przekształconego do postaci szeregowej.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IObjectMessage
```

.NET właściwości

Obiekt-pobierz i ustaw obiekt jako bajty

Interfejs:

```
System.Object Object
    {
      get;
      set;
    }

    Byte[] GetObject();
```

Pobierz i ustaw obiekt, który tworzy treść komunikatu obiektu.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException
- MessageNotWritableException

Dziedziczone właściwości i metody

Następujące właściwości zostały odziedziczone po interfejsie IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimdanym, JMSType, Właściwości

Następujące metody zostały odziedziczone po interfejsie IMessage:

clearBody, clearProperties, PropertyExists

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IPropertyContext

Interfejs IPropertyContext jest abstrakcyjną nadklasą zawierającą metody służące do pobierania i ustawiania właściwości. Metody te są dziedziczone przez inne klasy.

Hierarchia dziedziczenia:
Brak

metody

Właściwość GetBoolean-pobierz właściwość boolowską

Interfejs:

```
Boolean GetBooleanProperty(String property_name);
```

Pobierz wartość właściwości boolowskiej o podanej nazwie.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetByte-Pobieranie właściwości Byte

Interfejs:

```
Byte    GetByteProperty(String property_name) ;
Int16   GetSignedByteProperty(String property_name) ;
```

Pobieranie wartości właściwości bajtowej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetBytes-Pobieranie właściwości tablicy bajtów

Interfejs:

```
Byte[]  GetBytesProperty(String property_name) ;
```

Pobieranie wartości właściwości tablicy bajtów identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Liczba bajtów w tablicy.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetChar-pobranie właściwości znaku

Interfejs:

```
Char    GetCharProperty(String property_name) ;
```

Pobierz wartość 2-bajtowej właściwości znakowej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetDouble-Uzyskanie właściwości zmiennopozycyjnej o podwójnej precyzji

Interfejs:

```
Double  GetDoubleProperty(String property_name) ;
```

Pobieranie wartości właściwości zmiennopozycyjnej o podwójnej precyzji identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetFloat-pobranie właściwości zmiennopozycyjnej

Interfejs:

```
Single  GetFloatProperty(String property_name) ;
```

Pobieranie wartości właściwości zmiennopozycyjnej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetInt-Właściwość GetInt

Interfejs:

```
Int32   GetIntProperty(String property_name) ;
```

Pobieranie wartości właściwości będącej liczbą całkowitą identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetLong-Uzyskanie właściwości typu Long Integer

Interfejs:

```
Int64   GetLongProperty(String property_name) ;
```

Pobieranie wartości właściwości typu long integer identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetObject-Pobranie właściwości obiektu

Interfejs:

```
Object  GetObjectProperty( String property_name) ;
```

Pobieranie wartości i typu danych właściwości identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości, która jest jednym z następujących typów obiektów:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetShort-Pobranie właściwości typu Short Integer

Interfejs:

```
Int16   GetShortProperty(String property_name) ;
```

Pobieranie wartości właściwości typu short integer identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Wartość właściwości.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość GetString-właściwość GetString

Interfejs:

```
String  GetStringProperty(String property_name) ;
```

Pobierz wartość właściwości łańcuchowej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

Zwraca:
Obiekt typu String obudowujący łańcuch, który jest wartością właściwości. Jeśli konwersja danych jest wymagana, ta wartość jest łańcuchem po konwersji.

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException

Właściwość SetBoolean-ustaw właściwość boolowską

Interfejs:

```
void SetBooleanProperty( String property_name,  Boolean value) ;
```

Ustaw wartość właściwości boolowskiej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetByte-właściwość Set Byte

Interfejs:

```
void SetByteProperty( String property_name,  Byte value) ;
      void SetSignedByteProperty( String property_name,  Int16 value) ;
```

Ustaw wartość właściwości bajtowej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetBytes-ustawianie właściwości tablicy bajtów

Interfejs:

```
void SetBytesProperty( String property_name, Byte[] value ) ;
```

Ustaw wartość właściwości tablicy bajtów identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości, która jest tablicą bajtów.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetChar-właściwość Set Character

Interfejs:

```
void SetCharProperty( String property_name,  Char value) ;
```

Ustaw wartość 2-bajtowej właściwości znaku identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetDouble-ustawienie właściwości zmiennopozycyjnej o podwójnej precyzji

Interfejs:

```
void SetDoubleProperty( String property_name,  Double value) ;
```

Ustaw wartość właściwości zmiennopozycyjnej o podwójnej precyzji identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetFloat-ustawienie właściwości zmiennopozycyjnej

Interfejs:

```
void SetFloatProperty( String property_name,  Single value) ;
```

Ustaw wartość właściwości zmiennopozycyjnej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetInt-Ustaw właściwość Integer

Interfejs:

```
void SetIntProperty( String property_name,  Int32 value) ;
```

Ustaw wartość właściwości całkowitoliczbowej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetLong-Ustaw właściwość typu Long Integer

Interfejs:

```
void SetLongProperty( String property_name,  Int64 value) ;
```

Ustaw wartość właściwości typu long integer identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetObject-właściwość Set Object

Interfejs:

```
void SetObjectProperty( String property_name,  Object value) ;
```

Ustaw wartość i typ danych właściwości identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

objectType (dane wejściowe)
Wartość właściwości, która musi być jednym z następujących typów obiektów:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

wartość (dane wejściowe)
Wartość właściwości w postaci tablicy bajtów.

długość (wejście)
Liczba bajtów w tablicy.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetShort-Ustaw właściwość Short Integer

Interfejs:

```
void SetShortProperty( String property_name,  Int16 value) ;
```

Ustaw wartość właściwości short integer identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Wartość właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Właściwość SetString-właściwość Set String

Interfejs:

```
void SetStringProperty( String property_name,  String value);
```

Ustaw wartość właściwości łańcuchowej identyfikowanej przez nazwę.

Parametry:

nazwa_właściwości (wejście)
Obiekt typu String obudowujący nazwę właściwości.

wartość (dane wejściowe)
Obiekt typu String obudowujący łańcuch, który jest wartością właściwości.

Zwraca:
Unieważnione

Kontekst wątku:
Określona przez podklasę

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

IQueueBrowser

Aplikacja używa przeglądarki kolejek do przeglądania komunikatów w kolejce bez ich usuwania.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
System.Collections.IEnumerable
   |
   +----IBM.XMS.IQueueBrowser
```

.NET właściwości

MessageSelector -pobieranie selektora komunikatów

Interfejs:

```
String MessageSelector
{
  get;
}
```

Pobierz selektor komunikatów dla przeglądarki kolejek.

Selektor komunikatów jest obiektem typu String obudowującym wyrażenie selektora komunikatów. Jeśli konwersja danych jest wymagana, ta wartość jest wyrażeniem selektora komunikatów po konwersji. Jeśli przeglądarka kolejek nie ma selektora komunikatów, metoda zwraca obiekt typu String o wartości NULL.

Wyjątki:

- Wyjątek XMSException

Kolejka-kolejka pobierania

Interfejs:

```
IDestination Queue
{
  get;
}
```

Pobierz kolejkę powiązaną z przeglądarką kolejek jako obiekt docelowy reprezentujący kolejkę.

Wyjątki:

- Wyjątek XMSException

metody

Zamknij-zamknij przeglądarkę kolejek

Interfejs:

```
void  Close();
```

Zamknij przeglądarkę kolejek.

Jeśli aplikacja próbuje zamknąć przeglądarkę kolejek, która jest już zamknięta, wywołanie jest ignorowane.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

GetEnumerator -pobieranie komunikatów

Interfejs:

```
IEnumerator GetEnumerator();
```

Pobierz listę komunikatów w kolejce.

Ta metoda zwraca element wyliczeniowy, który hermetyzuje listę obiektów Message. Kolejność obiektów komunikatu jest taka sama, jak kolejność, w jakiej komunikaty będą pobierane z kolejki. Aplikacja może następnie użyć wyliczeniowego do przeglądania każdego komunikatu po kolei.

Element wyliczenia jest aktualizowany dynamicznie, gdy komunikaty są umieszczane w kolejce i usuwane z kolejki. Za każdym razem, gdy aplikacja wywołuje funkcję IEnumerator.MoveNext() w celu przeglądania następnego komunikatu w kolejce, komunikat odzwierciedla bieżącą zawartość kolejki.

Jeśli aplikacja wywoła tę metodę więcej niż raz dla przeglądarki kolejek, każde wywołanie zwraca nowy element wyliczeniowy. Dlatego aplikacja może używać więcej niż jednego enumeratora do przeglądania komunikatów w kolejce i utrzymywania wielu pozycji w kolejce.

Parametry:
Brak

Zwraca:
Obiekt iteratora.

Wyjątki:

- Wyjątek XMSException

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

Żądający

Aplikacja używa requestera do wysyłania komunikatu żądania, a następnie oczekiwania na odpowiedź i jej odbierania.

Hierarchia dziedziczenia:
Brak

Konstruktory

Zlecający-Utwórz zlecającego

Interfejs:

```
Requestor(ISession sess, IDestination dest);
```

Utwórz zamawiającego.

Parametry:

sess (wejście)
Obiekt sesji. Sesja nie może być transakcją i musi mieć jeden z następujących trybów potwierdzenia:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Dest (wejście)
Obiekt docelowy reprezentujący miejsce docelowe, do którego aplikacja może wysyłać komunikaty żądań.

Kontekst wątku:
Sesja powiązana z requesterem

Wyjątki:

- Wyjątek XMSException

metody

Zamknij-zamknij osobę żądającą

Interfejs:

```
void  Close();
```

Zamknij osobę żądającą.

Jeśli aplikacja podejmie próbę zamknięcia requestera, który jest już zamknięty, wywołanie zostanie zignorowane.

Uwaga: Gdy aplikacja zamyka osobę żądającą, powiązana sesja również nie jest zamykana. W tym względzie XMS zachowuje się inaczej niż JMS.

Parametry:
Brak

Zwraca:
Unieważnione

Kontekst wątku:
Dowolna

Wyjątki:

- Wyjątek XMSException

Żądanie-odpowiedź na żądanie

Interfejs:

```
IMessage Request(IMessage requestMessage);
```

Wyślij komunikat żądania, a następnie poczekaj i odbierz odpowiedź z aplikacji, która odbiera komunikat żądania.

Wywołanie tej metody blokuje się do momentu odebrania odpowiedzi lub do czasu zakończenia sesji, w zależności od tego, co nastąpi wcześniej.

Parametry:

requestMessage (wejście)
Obiekt Message obudowujący komunikat żądania.

Zwraca:
Wskaźnik do obiektu Message obudowującego komunikat odpowiedzi.

Kontekst wątku:
Sesja powiązana z requesterem

Wyjątki:

- Wyjątek XMSException

Wyjątek ResourceAllocation

Program XMS zgłasza ten wyjątek, jeśli program XMS nie może przydzielić zasobów wymaganych przez metodę.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.ResourceAllocationException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

SecurityException

Produkt XMS zgłasza ten wyjątek, jeśli identyfikator użytkownika i hasło podane w celu uwierzytelnienia aplikacji zostaną odrzucone. XMS zgłasza również ten wyjątek, jeśli sprawdzenie uprawnień nie powiedzie się i metoda nie zostanie zakończona.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.SecurityException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

ISesja

Sesja jest jednowątkowym kontekstem do wysyłania i odbierania komunikatów.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.ISession
```

Listę zdefiniowanych w systemie XMS właściwości obiektu sesji zawiera sekcja “Właściwości sesji” na stronie 2108.

.NET właściwości

AcknowledgeMode -tryb pobierania potwierdzenia

Interfejs:

```
AcknowledgeMode AcknowledgeMode
    {
      get;
    }
```

Pobierz tryb potwierdzenia dla sesji.

Tryb potwierdzenia jest określany podczas tworzenia sesji.

Jeśli sesja nie jest transakcją, tryb potwierdzenia może przyjmować jedną z następujących wartości:

`AcknowledgeMode.AutoAcknowledge`
`AcknowledgeMode.ClientAcknowledge`
`AcknowledgeMode.DupsOkAcknowledge`

Więcej informacji na temat trybów potwierdzania zawiera sekcja Potwierdzanie komunikatów.

Sesja, która jest transakcją, nie ma trybu potwierdzenia. Jeśli sesja jest transakcją, metoda zwraca wartość `AcknowledgeMode.SessionTransacted` .

Wyjątki:

- Wyjątek XMSException

Transakcyjne-określ, czy transakcyjne

Interfejs:

```
Boolean Transacted
    {
      get;
    }
```

Określ, czy sesja jest transakcją.

Dane transakcyjne są następujące:

- Wartość true, jeśli sesja jest transakcją.
- Fałsz, jeśli sesja nie jest transakcją.

W przypadku połączenia w czasie rzeczywistym z brokerem metoda zawsze zwraca wartość False.

Wyjątki:

- Wyjątek XMSException

metody

Zamknij-Zamknij sesję

Interfejs:

```
void Close();
```

Zamknij sesję. Jeśli sesja jest transakcyjna, wszystkie transakcje w toku są wycofywane.

Jeśli aplikacja próbuje zamknąć sesję, która jest już zamknięta, wywołanie jest ignorowane.

Parametry:
Brak

Zwraca:
Unieważnione

Kontekst wątku:
Dowolna

Wyjątki:

- Wyjątek XMSException

Zatwierdź-zatwierdź

Interfejs:

```
void Commit();
```

Zatwierdź wszystkie komunikaty przetworzone w bieżącej transakcji.

Sesja musi być sesją transakcją.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState
- TransactionRolledBackException

CreateBrowser -Tworzenie przeglądarki kolejek

Interfejs:

```
IQueueBrowser CreateBrowser(IDestination queue) ;
```

Utwórz przeglądarkę kolejek dla określonej kolejki.

Parametry:

kolejka (wejście)
Obiekt docelowy reprezentujący kolejkę.

Zwraca:
Obiekt QueueBrowser .

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination

CreateBrowser -Tworzenie przeglądarki kolejek (z selektorem komunikatów)

Interfejs:

```
IQueueBrowser CreateBrowser(IDestination queue, String selector) ;
```

Utwórz przeglądarkę kolejek dla określonej kolejki przy użyciu selektora komunikatów.

Parametry:

kolejka (wejście)
Obiekt docelowy reprezentujący kolejkę.

selektor (wejście)
Obiekt typu String obudowujący wyrażenie selektora komunikatów. Do przeglądarki kolejek są dostarczane tylko te komunikaty, które mają właściwości zgodne z wyrażeniem selektora komunikatów.

Pusty obiekt typu String oznacza, że nie ma selektora komunikatów dla przeglądarki kolejek.

Zwraca:
Obiekt QueueBrowser .

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination
- Wyjątek InvalidSelector

Komunikat CreateBytes-Komunikat tworzenia bajtów

Interfejs:

```
IBytesMessage CreateBytesMessage();
```

Utwórz komunikat bajtowy.

Parametry:
Brak

Zwraca:
Obiekt BytesMessage .

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState(sesja jest zamknięta)

CreateConsumer -Tworzenie konsumenta

Interfejs:

```
IMessageConsumer CreateConsumer(IDestination dest) ;
```

Utwórz konsument komunikatów dla określonego miejsca docelowego.

Parametry:

Dest (wejście)
Obiekt docelowy.

Zwraca:
Obiekt MessageConsumer .

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination

CreateConsumer -Utwórz konsument (z selektorem komunikatów)

Interfejs:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector) ;
```

Utwórz konsument komunikatów dla określonego miejsca docelowego przy użyciu selektora komunikatów.

Parametry:

Dest (wejście)
Obiekt docelowy.

selektor (wejście)
Obiekt typu String obudowujący wyrażenie selektora komunikatów. Do konsumenta komunikatów są dostarczane tylko te komunikaty, które mają właściwości zgodne z wyrażeniem selektora komunikatów.

Pusty obiekt typu String oznacza, że nie ma selektora komunikatów dla konsumenta komunikatów.

Zwraca:
Obiekt MessageConsumer .

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination
- Wyjątek InvalidSelector

CreateConsumer -Utwórz konsument (z selektorem komunikatów i flagą komunikatu lokalnego)

Interfejs:

```
IMessageConsumer CreateConsumer(IDestination dest,
                                String selector,
                                Boolean noLocal) ;
```

Utwórz konsument komunikatów dla określonego miejsca docelowego przy użyciu selektora komunikatów i, jeśli miejsce docelowe jest tematem, określ, czy konsument komunikatów odbiera komunikaty publikowane przez własne połączenie.

Parametry:

Dest (wejście)
Obiekt docelowy.

selektor (wejście)
Obiekt typu String obudowujący wyrażenie selektora komunikatów. Do konsumenta komunikatów są dostarczane tylko te komunikaty, które mają właściwości zgodne z wyrażeniem selektora komunikatów.

Pusty obiekt typu String oznacza, że nie ma selektora komunikatów dla konsumenta komunikatów.

noLocal (wejście)
Wartość True oznacza, że konsument komunikatów nie odbiera komunikatów opublikowanych przez własne połączenie. Wartość False oznacza, że konsument komunikatów odbiera komunikaty publikowane przez jego własne połączenie. Wartością domyślną jest false (fałsz).

Zwraca:
Obiekt MessageConsumer .

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination
- Wyjątek InvalidSelector

Subskrybent CreateDurable-Utwórz trwały subskrybent.

Interfejs:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription) ;
```

Utwórz trwałego subskrybenta dla określonego tematu.

Ta metoda nie jest poprawna dla połączenia z brokerem w czasie rzeczywistym.

Więcej informacji na temat trwałych subskrybentów zawiera sekcja Trwali subskrybenci.

Parametry:

Dest (wejście)
Obiekt docelowy reprezentujący temat. Temat nie może być tematem tymczasowym.

subskrypcja (wejście)
Obiekt typu String obudowujący nazwę, która identyfikuje trwałą subskrypcję. Nazwa musi być unikalna w obrębie identyfikatora klienta dla połączenia.

Zwraca:
Obiekt MessageConsumer reprezentujący trwałego subskrybenta.

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination

CreateDurableSubskrybent-Utwórz trwały subskrybent (z selektorem komunikatów i flagą komunikatu lokalnego)

Interfejs:

```
IMessageConsumer CreateDurableSubscriber(IDestination dest,
                                         String subscription,
                                         String selector,
                                         Boolean noLocal) ;
```

Utwórz trwały subskrybent dla określonego tematu przy użyciu selektora komunikatów i określ, czy trwały subskrybent odbiera komunikaty publikowane przez własne połączenie.

Ta metoda nie jest poprawna dla połączenia z brokerem w czasie rzeczywistym.

Więcej informacji na temat trwałych subskrybentów zawiera sekcja Trwali subskrybenci.

Parametry:

Dest (wejście)
Obiekt docelowy reprezentujący temat. Temat nie może być tematem tymczasowym.

subskrypcja (wejście)
Obiekt typu String obudowujący nazwę, która identyfikuje trwałą subskrypcję. Nazwa musi być unikalna w obrębie identyfikatora klienta dla połączenia.

selektor (wejście)
Obiekt typu String obudowujący wyrażenie selektora komunikatów. Do trwałego subskrybenta są dostarczane tylko te komunikaty, które mają właściwości zgodne z wyrażeniem selektora komunikatów.

Pusty obiekt typu String oznacza, że nie ma selektora komunikatów dla trwałego subskrybenta.

noLocal (wejście)
Wartość True oznacza, że trwały subskrybent nie odbiera komunikatów opublikowanych przez własne połączenie. Wartość False oznacza, że trwały subskrybent odbiera komunikaty publikowane przez własne połączenie. Wartością domyślną jest false (fałsz).

Zwraca:
Obiekt MessageConsumer reprezentujący trwałego subskrybenta.

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination
- Wyjątek InvalidSelector

Komunikat CreateMap-Utwórz komunikat odwzorowania

Interfejs:

```
IMapMessage CreateMapMessage();
```

Utwórz komunikat odwzorowania.

Parametry:
Brak

Zwraca:
Obiekt MapMessage .

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState(sesja jest zamknięta)

CreateMessage -Tworzenie komunikatu

Interfejs:

```
IMessage CreateMessage();
```

Utwórz komunikat, który nie ma treści.

Parametry:
Brak

Zwraca:
Obiekt Message.

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState(sesja jest zamknięta)

Komunikat CreateObject-Komunikat tworzenia obiektu

Interfejs:

```
IObjectMessage CreateObjectMessage();
```

Utwórz komunikat obiektu.

Parametry:
Brak

Zwraca:
Obiekt ObjectMessage .

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState(sesja jest zamknięta)

CreateProducer -Tworzenie producenta

Interfejs:

```
IMessageProducer CreateProducer(IDestination dest) ;
```

Utwórz producenta komunikatów w celu wysyłania komunikatów do określonego miejsca docelowego.

Parametry:

Dest (wejście)
Obiekt docelowy.

Jeśli zostanie podany obiekt miejsca docelowego o wartości NULL, producent komunikatów zostanie utworzony bez miejsca docelowego. W takim przypadku aplikacja musi określić miejsce docelowe za każdym razem, gdy używa producenta komunikatów do wysłania komunikatu.

Zwraca:
Obiekt MessageProducer .

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination

CreateQueue -Tworzenie kolejki

Interfejs:

```
IDestination CreateQueue(String queue) ;
```

Utwórz obiekt docelowy, który będzie reprezentował kolejkę na serwerze przesyłania komunikatów.

Ta metoda nie tworzy kolejki na serwerze przesyłania komunikatów. Przed wywołaniem tej metody przez aplikację należy utworzyć kolejkę.

Parametry:

kolejka (wejście)
Obiekt łańcuchowy hermetyzujący nazwę kolejki lub hermetyzujący identyfikator URI (Uniform Resource Identifier) identyfikujący kolejkę.

Zwraca:
Obiekt docelowy reprezentujący kolejkę.

Wyjątki:

- Wyjątek XMSException

Komunikat CreateStream-Tworzenie komunikatu strumienia

Interfejs:

```
IStreamMessage CreateStreamMessage();
```

Utwórz komunikat strumienia.

Parametry:
Brak

Zwraca:
Obiekt StreamMessage .

Wyjątki:

- Wyjątek XMSException
- XMS_ILAL_STATE_EXCEPTION (XMS_ILLEGAL_STATE_EXCEPTION)

Kolejka CreateTemporary-Utwórz kolejkę tymczasową

Interfejs:

```
IDestination CreateTemporaryQueue() ;
```

Utwórz kolejkę tymczasową.

Zasięgiem kolejki tymczasowej jest połączenie. Kolejka tymczasowa może być używana tylko przez sesje utworzone przez połączenie.

Kolejka tymczasowa pozostaje do momentu jej jawnego usunięcia lub zakończenia połączenia, w zależności od tego, co nastąpi wcześniej.

Więcej informacji na temat kolejek tymczasowych zawiera sekcja Tymczasowe miejsca docelowe.

Parametry:
Brak

Zwraca:
Obiekt docelowy reprezentujący kolejkę tymczasową.

Wyjątki:

- Wyjątek XMSException

Temat CreateTemporary-Tworzenie tematu tymczasowego

Interfejs:

```
IDestination CreateTemporaryTopic() ;
```

Utwórz temat tymczasowy.

Zasięgiem tematu tymczasowego jest połączenie. Tylko sesje utworzone przez połączenie mogą korzystać z tematu tymczasowego.

Temat tymczasowy pozostaje do momentu jawnego usunięcia lub zakończenia połączenia, w zależności od tego, co nastąpi wcześniej.

Więcej informacji na temat tematów tymczasowych zawiera sekcja Tymczasowe miejsca docelowe.

Parametry:
Brak

Zwraca:
Obiekt docelowy reprezentujący temat tymczasowy.

Wyjątki:

- Wyjątek XMSException

Komunikat CreateText-Utwórz komunikat tekstowy

Interfejs:

```
ITextMessage CreateTextMessage();
```

Utwórz wiadomość tekstową z pustą treścią.

Parametry:
Brak

Zwraca:
Obiekt TextMessage .

Wyjątki:

- Wyjątek XMSException

Komunikat CreateText-Utwórz komunikat tekstowy (zainicjowany)

Interfejs:

```
ITextMessage CreateTextMessage(String initialValue);
```

Utwórz komunikat tekstowy, którego treść jest inicjowana określonym tekstem.

Parametry:

initialValue (wejście)
Obiekt typu String obudowujący tekst w celu zainicjowania treści komunikatu tekstowego.

Brak

Zwraca:
Obiekt TextMessage .

Wyjątki:

- Wyjątek XMSException

CreateTopic -Tworzenie tematu

Interfejs:

```
IDestination CreateTopic(String topic) ;
```

Utwórz obiekt docelowy do reprezentowania tematu.

Parametry:

topic (wejście)
Obiekt łańcuchowy hermetyzujący nazwę tematu lub hermetyzujący identyfikator URI (Uniform Resource Identifier) identyfikujący temat.

Zwraca:
Obiekt docelowy reprezentujący temat.

Wyjątki:

- Wyjątek XMSException

Odzyskiwanie-odzyskiwanie

Interfejs:

```
void Recover();
```

Odtwórz sesję. Dostarczanie komunikatów jest zatrzymywane, a następnie restartowane z najstarszym niepotwierdzonym komunikatem.

Sesja nie może być sesją transakcją.

Więcej informacji na temat odtwarzania sesji zawiera sekcja Potwierdzenie komunikatu.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek IllegalState

Wycofanie zmian-wycofanie zmian

Interfejs:

```
void Rollback();
```

Wycofaj wszystkie komunikaty przetworzone w bieżącej transakcji.

Sesja musi być sesją transakcją.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

- Wyjątek IllegalState

Anuluj subskrypcję-anuluj subskrypcję

Interfejs:

```
void Unsubscribe(String subscription);
```

Usuń trwałą subskrypcję. Serwer przesyłania komunikatów usuwa rekord trwałej subskrypcji, która jest przez niego utrzymywana, i nie wysyła więcej komunikatów do trwałego subskrybenta.

Aplikacja nie może usunąć trwałej subskrypcji w następujących okolicznościach:

- Gdy istnieje aktywny konsument komunikatów dla trwałej subskrypcji
- Gdy pobierany komunikat jest częścią transakcji oczekującej
- Podczas gdy zużyty komunikat nie został potwierdzony

Ta metoda nie jest poprawna dla połączenia z brokerem w czasie rzeczywistym.

Parametry:

subskrypcja (wejście)
Obiekt typu String zawierający nazwę identyfikującą trwałą subskrypcję.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- Wyjątek InvalidDestination
- Wyjątek IllegalState

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

IStreamMessage

Komunikat strumienia to komunikat, którego treść składa się ze strumienia wartości, z którym każda wartość ma powiązany typ danych. Treść jest zapisywana i odczytywana sekwencyjnie.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.IStreamMessage
```

Gdy aplikacja odczytuje wartość ze strumienia komunikatów, wartość może zostać przekształcona przez program XMS w inny typ danych. Więcej informacji na temat tej formy niejawnej konwersji zawiera sekcja Treść komunikatu XMS.

metody

ReadBoolean -wartość boolowska odczytu

Interfejs:

```
Boolean ReadBoolean();
```

Odczytaj wartość boolowską ze strumienia komunikatów.

Parametry:
Brak

Zwraca:
Wartość boolowska, która jest odczytywana.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadByte -Odczyt Byte

Interfejs:

```
Int16   ReadSignedByte();
    Byte    ReadByte();
```

Odczytaj ze strumienia komunikatów 8-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytany bajt.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadBytes -Odczytane Bajty

Interfejs:

```
Int32   ReadBytes(Byte[] array);
```

Odczytaj tablicę bajtów ze strumienia komunikatów.

Parametry:

array (wejście)
Bufor zawierający tablicę odczytanych bajtów i długość buforu w bajtach.

Jeśli liczba bajtów w tablicy jest mniejsza lub równa długości buforu, do buforu jest wczytywana cała tablica. Jeśli liczba bajtów w tablicy jest większa niż długość buforu, bufor jest wypełniany częścią tablicy, a kursor wewnętrzny oznacza pozycję następnego bajtu, który ma zostać odczytany. Kolejne wywołanie metody readBytes() powoduje odczytanie bajtów z tablicy począwszy od bieżącej pozycji kursora.

Jeśli na wejściu zostanie podany wskaźnik pusty, wywołanie pominie tablicę bajtów bez jej odczytu.

Zwraca:
Liczba bajtów, które zostały wczytane do buforu. Jeśli bufor jest częściowo zapełniony, wartość jest mniejsza niż długość buforu, co oznacza, że w tablicy nie ma więcej bajtów do odczytania. Jeśli przed wywołaniem nie ma już żadnych bajtów do odczytania z tablicy, wartością jest XMSC_END_OF_BYTEARRAY.

Jeśli na wejściu zostanie określony wskaźnik pusty, metoda nie zwróci żadnej wartości.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadChar -Odczyt znaku

Interfejs:

```
Char    ReadChar();
```

Odczytaj 2-bajtowy znak ze strumienia komunikatów.

Parametry:
Brak

Zwraca:
Odczytany znak.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadDouble -odczyt liczby zmiennopozycyjnej o podwójnej precyzji

Interfejs:

```
Double  ReadDouble();
```

Odczytaj 8-bajtową liczbę zmiennopozycyjną o podwójnej precyzji ze strumienia komunikatów.

Parametry:
Brak

Zwraca:
Liczba zmiennopozycyjna o podwójnej precyzji, która jest odczytywana.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadFloat -odczyt liczby zmiennopozycyjnej

Interfejs:

```
Single  ReadFloat();
```

Odczytaj 4-bajtową liczbę zmiennopozycyjną ze strumienia komunikatów.

Parametry:
Brak

Zwraca:
Liczba zmiennopozycyjna, która jest odczytywana.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadInt -Odczyt liczby całkowitej

Interfejs:

```
Int32   ReadInt();
```

Odczytaj ze strumienia komunikatów 32-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytana liczba całkowita.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadLong -Odczyt Long Integer

Interfejs:

```
Int64   ReadLong();
```

Odczytaj 64-bitową liczbę całkowitą ze znakiem ze strumienia komunikatów.

Parametry:
Brak

Zwraca:
Odczytana liczba całkowita typu long.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadObject -odczyt obiektu

Interfejs:

```
Object  ReadObject();
```

Odczytaj wartość ze strumienia komunikatów i zwróć jej typ danych.

Parametry:
Brak

Zwraca:
Wartość, która jest jednym z następujących typów obiektów:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

Wyjątki:
Wyjątek XMSException

ReadShort -Odczyt Short Integer (krótka liczba całkowita)

Interfejs:

```
Int16   ReadShort();
```

Odczytaj ze strumienia komunikatów 16-bitową liczbę całkowitą ze znakiem.

Parametry:
Brak

Zwraca:
Odczytana krótka liczba całkowita.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

ReadString -odczytany łańcuch

Interfejs:

```
String  ReadString();
```

Odczytaj łańcuch ze strumienia komunikatów. Jeśli jest to wymagane, program XMS przekształca znaki w łańcuchu w lokalną stronę kodową.

Parametry:
Brak

Zwraca:
Obiekt typu String obudowujący odczytany łańcuch. Jeśli konwersja danych jest wymagana, jest to łańcuch po konwersji.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

Resetuj-Resetuj

Interfejs:

```
void  Reset();
```

Przełącz treść komunikatu w tryb tylko do odczytu i zmień pozycję kursora na początku strumienia komunikatów.

Parametry:
Brak

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException
- MessageEOFException

WriteBoolean -wartość boolowska zapisu

Interfejs:

```
void  WriteBoolean(Boolean value);
```

Zapisz wartość boolowską w strumieniu komunikatów.

Parametry:

wartość (dane wejściowe)
Wartość boolowska, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteByte -Zapis w bajtach

Interfejs:

```
void  WriteByte(Byte value);
void  WriteSignedByte(Int16 value);
```

Zapisz bajt w strumieniu komunikatów.

Parametry:

wartość (dane wejściowe)
Bajt, który ma zostać zapisany.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteBytes -Zapisane bajty (Write Bytes)

Interfejs:

```
void  WriteBytes(Byte[] value);
```

Zapisz tablicę bajtów w strumieniu komunikatów.

Parametry:

wartość (dane wejściowe)
Tablica bajtów do zapisania.

długość (wejście)
Liczba bajtów w tablicy.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteChar -Zapisz znak

Interfejs:

```
void  WriteChar(Char value);
```

Zapisz znak w strumieniu komunikatów jako 2 bajty, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Znak, który ma zostać zapisany.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteDouble -liczba zmiennopozycyjna o podwójnej precyzji dla zapisu

Interfejs:

```
void  WriteDouble(Double value);
```

Przekształć liczbę zmiennopozycyjną o podwójnej precyzji w długą liczbę całkowitą i zapisz długą liczbę całkowitą w strumieniu komunikatów jako 8 bajtów (na początku najstarszy bajt).

Parametry:

wartość (dane wejściowe)
Liczba zmiennopozycyjna podwójnej precyzji, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteFloat -liczba zmiennopozycyjna zapisu

Interfejs:

```
void  WriteFloat(Single value);
```

Przekształć liczbę zmiennopozycyjną w liczbę całkowitą i zapisz ją w strumieniu komunikatów jako 4 bajty (najpierw najstarszy bajt).

Parametry:

wartość (dane wejściowe)
Liczba zmiennopozycyjna, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteInt -Write Integer (liczba całkowita)

Interfejs:

```
void  WriteInt(Int32 value);
```

Zapisz liczbę całkowitą w strumieniu komunikatów jako 4 bajty, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Liczba całkowita do zapisania.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteLong -Zapis Long Integer

Interfejs:

```
void  WriteLong(Int64 value);
```

W strumieniu komunikatów należy zapisać długą liczbę całkowitą jako 8 bajtów, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Liczba całkowita typu long, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteObject -zapis obiektu

Interfejs:

```
void  WriteObject(Object value);
```

Zapisz wartość o określonym typie danych w strumieniu komunikatów.

Parametry:

objectType (dane wejściowe)
Wartość, która musi być jednym z następujących typów obiektów:

```
Boolean
Byte
Byte[]
Char
Double
Single
Int32
Int64
Int16
String
```

wartość (dane wejściowe)
Tablica bajtów zawierająca wartość, która ma zostać zapisana.

długość (wejście)
Liczba bajtów w tablicy.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException

WriteShort -Short Write Integer (krótka liczba całkowita)

Interfejs:

```
void  WriteShort(Int16 value);
```

Zapisz krótką liczbę całkowitą w strumieniu komunikatów jako 2 bajty, najpierw najstarszy bajt.

Parametry:

wartość (dane wejściowe)
Krótka liczba całkowita, która ma zostać zapisana.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

WriteString -zapis łańcucha

Interfejs:

```
void  WriteString(String value);
```

Zapisz łańcuch w strumieniu komunikatów.

Parametry:

wartość (dane wejściowe)
Obiekt typu String obudowujący łańcuch, który ma zostać zapisany.

Zwraca:
Unieważnione

Wyjątki:

- Wyjątek XMSException
- MessageNotWritableException

Dziedziczone właściwości i metody

Następujące właściwości zostały odziedziczone po interfejsie IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimdanym, JMSType, Właściwości

Następujące metody zostały odziedziczone po interfejsie IMessage:

clearBody, clearProperties, PropertyExists

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

ITextMessage

Komunikat tekstowy to komunikat, którego treść składa się z łańcucha.

Hierarchia dziedziczenia:

```
IBM.XMS.IPropertyContext
   |
   +----IBM.XMS.IMessage
           |
           +----IBM.XMS.ITextMessage
```

.NET właściwości

Tekst-Pobierz i ustaw tekst

Interfejs:

```
String Text
{
  get;
  set;
}
```

Pobierz i ustaw łańcuch, który tworzy treść komunikatu tekstowego.

Jeśli jest to wymagane, program XMS przekształca znaki w łańcuchu w lokalną stronę kodową.

Wyjątki:

- Wyjątek XMSException
- MessageNotReadableException

- MessageNotWritableException
- MessageEOFException

Dziedziczone właściwości i metody

Następujące właściwości zostały odziedziczone po interfejsie IMessage:

JMSCorrelationID, JMSDeliveryMode, JMSDestination, JMSExpiration, JMSMessageID, JMSPriority, JMSRedelivered, JMSReplyTo, JMSTimdanym, JMSType, Właściwości

Następujące metody zostały odziedziczone po interfejsie IMessage:

clearBody, clearProperties, PropertyExists

Następujące metody zostały odziedziczone po interfejsie IPropertyContext:

GetBooleanProperty, GetByteProperty, GetBytesProperty, GetCharProperty, GetDoubleProperty, GetFloatProperty, GetIntProperty, GetLongProperty, GetObjectProperty, GetShortProperty, GetStringProperty, SetBooleanProperty, SetByteProperty, SetBytesProperty, SetCharProperty, SetDoubleProperty, SetFloatProperty, SetIntProperty, SetLongProperty, SetObjectProperty, SetShortProperty, SetStringProperty

TransactionInProgressException

Program XMS zgłasza ten wyjątek, jeśli aplikacja żąda operacji, która nie jest poprawna, ponieważ transakcja jest w toku.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionInProgressException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

TransactionRolledBackException

XMS zgłasza ten wyjątek, jeśli aplikacja wywołuje funkcję Session.commit() w celu zatwierdzenia bieżącej transakcji, ale transakcja jest wycofywana.

Hierarchia dziedziczenia:

```
IBM.XMS.XMSException
   |
   +----IBM.XMS.XMSException
           |
           +----IBM.XMS.TransactionRolledBackException
```

Dziedziczone właściwości i metody

Następujące metody zostały odziedziczone po interfejsie XMSException:

KodGetError, wyjątek GetLinked

Wyjątek XMSException

Jeśli program XMS wykryje błąd podczas przetwarzania wywołania metody .NET , program XMS zgłosi wyjątek. Wyjątkiem jest obiekt, który hermetyzuje informacje o błędzie.

Hierarchia dziedziczenia:

```
System.Exception
   |
   +----IBM.XMS.XMSException
```

Istnieją różne typy wyjątków XMS , a obiekt XMSException jest tylko jednym z typów wyjątków. Jednak klasa XMSException jest nadklasą innych klas wyjątków XMS . XMS zgłasza wyjątek XMSException w sytuacjach, w których żaden inny typ wyjątku nie jest odpowiedni.

.NET właściwości

ErrorCode -uzyskaj kod błędu

Interfejs:

```
public String ErrorCode
{
  get {return errorCode_;}
}
```

Uzyskaj kod błędu.

Wyjątki:

- Wyjątek XMSException

LinkedException -wyjątek pobierania połączenia

Interfejs:

```
public Exception LinkedException
{
  get { return linkedException_;}
  set { linkedException_ = value;}
}
```

Pobierz następny wyjątek w łańcuchu wyjątków.

Metoda zwraca wartość NULL, jeśli w łańcuchu nie ma więcej wyjątków.

Wyjątki:

- Wyjątek XMSException

XMSFactoryFactory

Jeśli aplikacja nie używa obiektów administrowanych, należy użyć tej klasy do tworzenia fabryk połączeń, kolejek i tematów.

Hierarchia dziedziczenia:
Brak

.NET właściwości

Metadane-pobieranie metadanych

Interfejs:

```
IConnectionMetaData MetaData
```

Pobierz metadane odpowiednie dla typu połączenia obiektu XMSFactoryFactory .

Wyjątki:
Brak

metody

Fabryka CreateConnection-Utwórz fabrykę połączeń

Interfejs:

```
IConnectionFactory CreateConnectionFactory();
```

Utwórz obiekt ConnectionFactory zadeklarowanego typu.

Parametry:
Brak

Zwraca:
Obiekt ConnectionFactory .

Wyjątki:

- Wyjątek XMSException

CreateQueue -Tworzenie kolejki

Interfejs:

```
IDestination CreateQueue(String name);
```

Utwórz obiekt docelowy, który będzie reprezentował kolejkę na serwerze przesyłania komunikatów.

Ta metoda nie tworzy kolejki na serwerze przesyłania komunikatów. Przed wywołaniem tej metody przez aplikację należy utworzyć kolejkę.

Parametry:

nazwa (wejście)
Obiekt łańcuchowy hermetyzujący nazwę kolejki lub hermetyzujący identyfikator URI (Uniform Resource Identifier) identyfikujący kolejkę.

Zwraca:
Obiekt docelowy reprezentujący kolejkę.

Wyjątki:

- Wyjątek XMSException

CreateTopic -Tworzenie tematu

Interfejs:

```
IDestination CreateTopic(String name);
```

Utwórz obiekt docelowy do reprezentowania tematu.

Parametry:

nazwa (wejście)
Obiekt łańcuchowy hermetyzujący nazwę tematu lub hermetyzujący identyfikator URI (Uniform Resource Identifier) identyfikujący temat.

Zwraca:
Obiekt docelowy reprezentujący temat.

Wyjątki:

- Wyjątek XMSException

GetInstance -pobranie instancji klasy XMSFactoryFactory

Interfejs:

```
static XMSFactoryFactory GetInstance(int connectionType);
```

Utwórz instancję klasy XMSFactoryFactory. Aplikacja XMS używa obiektu XMSFactoryFactory do uzyskania odwołania do obiektu ConnectionFactory , który jest odpowiedni dla wymaganego typu protokołu. Ten obiekt ConnectionFactory może następnie generować połączenia tylko dla tego typu protokołu.

Parametry:

connectionType (wejście)
Typ połączenia, dla którego obiekt ConnectionFactory tworzy połączenia:

- XMSC.CT_WPM
- XMSC.CT_RTT
- XMSC.CT_WMQ

Zwraca:
Obiekt XMSFactoryFactory dedykowany dla zadeklarowanego typu połączenia.

Wyjątki:

- Wyjątek NotSupported

Właściwości obiektów XMS

W tej sekcji opisano właściwości obiektu zdefiniowane przez XMS.

Ta sekcja zawiera informacje o następujących typach obiektów:

- "Właściwości połączenia" na stronie 2094
- "Właściwości obiektu ConnectionFactory" na stronie 2094
- "Właściwości danych ConnectionMeta" na stronie 2099
- "Właściwości miejsca docelowego" na stronie 2100
- "Właściwości kontekstu InitialContext" na stronie 2101
- "Właściwości komunikatu" na stronie 2102
- "Właściwości elementu MessageConsumer" na stronie 2107
- "Właściwości producenta MessageProducer" na stronie 2108
- "Właściwości sesji" na stronie 2108

Opis każdego typu obiektu zawiera listę właściwości obiektu określonego typu oraz krótki opis każdej właściwości.

Ta sekcja zawiera również definicję każdej właściwości (patrz sekcja "Definicje właściwości" na stronie 2108).

Jeśli aplikacja definiuje własne właściwości obiektów opisanych w tej sekcji, nie powoduje błędu, ale może spowodować nieprzewidywalne rezultaty.

Uwaga: Nazwy i wartości właściwości w tej sekcji są wyświetlane w postaci XMSC.*NAME*, która jest formularzem używanym w językach C i C + +. Jednak w języku .NETnazwa właściwości może mieć postać XMSC.*NAME* lub XMSC_*NAME*, w zależności od sposobu jej używania:

- W przypadku określania właściwości nazwa właściwości musi mieć postać XMSC.*NAME* , jak pokazano w poniższym przykładzie:

```
cf.SetStringProperty(XMSC.WMQ_CHANNEL, "DOTNET.SVRCONN");
```

- W przypadku podawania łańcucha nazwa właściwości musi mieć postać XMSC_*NAME* , jak pokazano w poniższym przykładzie:

```
cf.SetStringProperty("XMSC_WMQ_CHANNEL", "DOTNET.SVRCONN");
```

  W produkcie .NETnazwy i wartości właściwości są dostarczane jako stałe w klasie XMSC. Stałe te identyfikują łańcuchy i będą używane przez dowolną aplikację XMS .NET . Jeśli używane są te predefiniowane stałe, nazwy i wartości właściwości mają postać XMSC.*NAZWA*, więc na przykład można użyć nazwy XMSC.USERID, a nie XMSC_USERID.

Typy danych są również w formie używanej dla języków C/C + +. Odpowiednie wartości dla języka .NET można znaleźć w sekcji Typy danych dla języka .NET.

Właściwości połączenia

Przegląd właściwości obiektu połączenia z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 872. Właściwości połączenia

Nazwa właściwości	**Opis**
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER” na stronie 2143	Ta właściwość jest używana do uzyskiwania nazwy menedżera kolejek, z którym jest połączony.
“XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID” na stronie 2143	Ta właściwość jest zapełniana identyfikatorem menedżera kolejek po nawiązaniu połączenia.
XMSC_WPM_CONNECTION_PROTOCOL	Protokół komunikacyjny używany na potrzeby połączenia z mechanizmem przesyłania komunikatów. Ta właściwość jest tylko do odczytu.
XMSC_WPM_HOST_NAME	Położenie mechanizmu przesyłania komunikatów, z którym połączona jest aplikacja (nazwa hosta lub adres IP systemu). Ta właściwość jest tylko do odczytu.
XMSC_WPM_ME_NAME	Nazwa mechanizmu przesyłania komunikatów, z którym połączona jest aplikacja. Ta właściwość jest tylko do odczytu.
XMSC_WPM_PORT	Numer portu, na którym nasłuchuje mechanizm przesyłania komunikatów, z którym aplikacja nawiązała połączenie. Ta właściwość jest tylko do odczytu.

Obiekt połączenia ma również właściwości tylko do odczytu, które pochodzą z właściwości fabryki połączeń użytej do utworzenia połączenia. Te właściwości pochodzą nie tylko z właściwości fabryki połączeń, które zostały ustawione w czasie tworzenia połączenia, ale także z wartości domyślnych właściwości, które nie zostały ustawione. Właściwości obejmują tylko te, które są odpowiednie dla typu serwera przesyłania komunikatów, z którym połączona jest aplikacja. Nazwy właściwości są takie same jak nazwy właściwości fabryki połączeń.

Właściwości obiektu ConnectionFactory

Przegląd właściwości obiektu ConnectionFactory z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 873. Właściwości obiektu ConnectionFactory	
Nazwa właściwości	**Opis**
“XMSC_ASYNC_EXCEPTIONS” na stronie 2118	Ta właściwość określa, czy aplikacja XMS ma informować interfejs ExceptionListener tylko wtedy, gdy połączenie zostało zerwane, czy też wtedy, gdy dowolny wyjątek wystąpi asynchronicznie w wywołaniu interfejsu API XMS. Ta właściwość ma zastosowanie względem wszystkich połączeń utworzonych w tej fabryce połączeń, które mają zarejestrowany interfejs ExceptionListener.
V 9.3.0 “XMSC_WMQ_BALANCING_APPLICATION_TYPE” na stronie 2127	Typ opcji równoważenia
V 9.3.0 “XMSC_WMQ_BALANCING_OPCJE” na stronie 2127	Opcje równoważenia ustawione przez aplikację wydającą
V 9.3.0 “XMSC_WMQ_BALANCING_TIMEOUT (XMSC_W_BALANCING_TIMEOUT)” na stronie 2128	Limit czasu, po upływie którego ponowne równoważenie może przerwać działanie aplikacji.
XMSC_CLIENT_ID	Identyfikator klienta dla połączenia.
XMSC_CONNECTION_TYPE	Typ serwera przesyłania komunikatów, z którym łączy się aplikacja.
HASŁO XMSC_PASSWORD	Hasło, które może być używane do uwierzytelniania aplikacji na etapie podejmowania próby nawiązania połączenia z serwerem przesyłania komunikatów.
“XMSC_RTT_BROKER_PING_INTERVAL” na stronie 2123	Odstęp czasu (w milisekundach), po upływie którego klient XMS .NET sprawdza połączenie z serwerem przesyłania komunikatów w czasie rzeczywistym w celu wykrycia dowolnego działania.
XMSC_RTT_CONNECTION_PROTOCOL	Protokół komunikacyjny używany do nawiązywania połączenia z brokerem w czasie rzeczywistym.
XMSC_RTT_HOST_NAME	Miejsce działania brokera: nazwa hosta lub adres IP systemu.
XMSC_RTT_LOCAL_ADDRESS	Nazwa hosta lub adres IP interfejsu sieci lokalnej na potrzeby nawiązania połączenia z brokerem w czasie rzeczywistym.
XMSC_RTT_MULTICAST (XMSC_RTT_MULTICAST)	Ustawienie rozsyłania grupowego dla fabryki połączeń lub miejsca docelowego.
XMSC_RTT_PORT	Numer portu, na którym broker nasłuchuje żądań przychodzących.
XMSC_USERID (XMSC_USERID)	Identyfikator użytkownika, który może być używany do uwierzytelniania aplikacji na etapie podejmowania próby nawiązania połączenia z serwerem przesyłania komunikatów.
XMSC_WMQ_BROKER_CONTROLQ	Nazwa kolejki sterującej używanej przez broker.
XMSC_WMQ_BROKER_PUBQ	Nazwa kolejki monitorowanej przez broker, do której aplikacje wysyłają publikowane komunikaty.

Tabela 873. Właściwości obiektu ConnectionFactory (kontynuacja)	
Nazwa właściwości	**Opis**
XMSC_WMQ_BROKER_QMGR	Nazwa menedżera kolejek, z którym broker nawiązał połączenie.
XMSC_WMQ_BROKER_SUBQ	Nazwa kolejki subskrybenta dla konsumenta nietrwałych komunikatów.
XMSC_WMQ_BROKER_VERSION	Typ brokera używany na potrzeby połączenia lub miejsca docelowego.
"XMSC_WMQ_CCDTURL" na stronie 2130	Adres URL identyfikujący nazwę i położenie pliku zawierającego tabelę definicji kanałów klienta oraz określający sposób dostępu do pliku.
XMSC_WMQ_CHANNEL	Nazwa kanału używanego do nawiązywania połączenia.
"XMSC_WMQ_CLIENT_RECONNECT_OPTIONS" na stronie 2131	Ta właściwość określa opcje ponownego połączenia klienta dla nowych połączeń utworzonych przez tę fabrykę
"XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT," na stronie 2131	Ta właściwość określa czas (w sekundach), przez jaki połączenie klienta próbuje ponownie nawiązać połączenie.
XMSC_WMQ_CONNECTION_MODE	Tryb, przy użyciu którego aplikacja nawiązuje połączenie z menedżerem kolejek.
"XMSC_WMQ_CONNECTION_NAME_LIST" na stronie 2132	Ta właściwość określa hosty, z którymi klient próbuje ponownie nawiązać połączenie po zerwaniu połączenia.
XMSC_WMQ_FAIL_IF_QUIESCE	Określa, czy wywołania pewnych metod nie powiodą się, jeśli menedżer kolejek, z którym aplikacja nawiązała połączenie, będzie w stanie wyciszania.
XMSC_WMQ_HOST_NAME	Miejsce działania menedżera kolejek: nazwa hosta lub adres IP systemu.
XMSC_WMQ_LOCAL_ADDRESS	W przypadku połączenia z menedżerem kolejek: ta właściwość określa używaną wartość interfejsu sieci lokalnej albo portu lokalnego (lub zakresu portów lokalnych) bądź obie te wartości.
XMSC_WMQ_MESSAGE_SELECTION	Określa, czy wybór komunikatów jest dokonywany przez klienta XMS , czy przez broker.
XMSC_WMQ_MSG_BATCH_SIZE	Maksymalna liczba komunikatów pobieranych z kolejki w jednej partii, gdy komunikaty są dostarczane w trybie asynchronicznym.
XMSC_WMQ_POLLING_INTERVAL	Jeśli kolejka żadnego procesu nasłuchującego w ramach sesji nie zawiera odpowiedniego komunikatu, jest to maksymalny odstęp czasu (w milisekundach) między kolejnymi próbami pobrania komunikatu z kolejki przez każdy z procesów nasłuchujących komunikatów.
"XMSC_WMQ_PROVIDER_VERSION" na stronie 2141	Wersja, wydanie, poziom modyfikacji i pakiet poprawek menedżera kolejek, z którym aplikacja ma nawiązać połączenie.
XMSC_WMQ_PORT	Numer portu, na którym menedżer kolejek nasłuchuje żądań przychodzących.
XMSC_WMQ_PUB_ACK_INTERVAL	Liczba komunikatów opublikowanych przez publikator, zanim klient XMS zażąda potwierdzenia od brokera.

Tabela 873. Właściwości obiektu ConnectionFactory (kontynuacja)	
Nazwa właściwości	**Opis**
“XMSC_WMQ_PUT_ASYNC_ALLOWED” na stronie 2136	Ta właściwość określa, czy producenci komunikatów mogą używać asynchronicznych operacji put w celu wysyłania komunikatów do tego miejsca docelowego.
XMSC_WMQ_QMGR_CCSID	Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, w którym pola danych znakowych zdefiniowane w interfejsie MQI (Message Queue Interface) są wymieniane między klientem XMS a klientem IBM MQ .
XMSC_WMQ_QUEUE_MANAGER	Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie.
XMSC_WMQ_RECEIVE_EXIT	Identyfikuje wyjście odbierania kanału, które ma zostać uruchomione.
XMSC_WMQ_RECEIVE_EXIT_INIT	Dane użytkownika przekazywane do wyjścia odbierania kanału w momencie jego wywołania.
XMSC_WMQ_SECURITY_EXIT	Identyfikuje wyjście zabezpieczeń kanału.
XMSC_WMQ_SECURITY_EXIT_INIT	Dane użytkownika przekazywane do wyjścia zabezpieczeń kanału w momencie jego wywołania.
“XMSC_WMQ_SEND_CHECK_COUNT” na stronie 2145	Liczba wywołań wysyłania dozwolonych między sprawdzeniami błędów asynchronicznego umieszczania w ramach jednej sesji XMS bez transakcji.
XMSC_WMQ_SEND_EXIT	Identyfikuje wyjście wysyłania kanału.
XMSC_WMQ_SEND_EXIT_INIT	Dane użytkownika przekazywane do wyjść wysyłania kanału w momencie ich wywołania.
“XMSC_WMQ_SHARE_CONV_ALLOWED” na stronie 2145	Określa, czy połączenie klienta może współużytkować gniazdo z innymi XMS połączeniami najwyższego poziomu z tego samego procesu do tego samego menedżera kolejek, jeśli definicje kanału są zgodne. Ta właściwość umożliwia pełną izolację połączeń w osobnych gniazdach (jeśli jest to konieczne z powodów związanych z tworzeniem, konserwacją lub działaniem aplikacji).
XMSC_WMQ_SSL_CERT_STORES	Lokalizacje serwerów, na których znajdują się listy odwołań certyfikatów (Certificate Revocation List – CRL) używane podczas nawiązywania połączenia SSL z menedżerem kolejek.
XMSC_WMQ_SSL_CIPHER_SPEC	Nazwa specyfikacji szyfrowania używanej w przypadku bezpiecznego połączenia z menedżerem kolejek.
XMSC_WMQ_SSL_CIPHER_SUITE	Nazwa zestawu algorytmów szyfrowania używanego w przypadku połączenia TLS z menedżerem kolejek. Na podstawie podanego zestawu algorytmów szyfrowania jest określany protokół używany do negocjowania bezpiecznego połączenia.
XMSC_WMQ_SSL_CRYPTO_HW	Szczegóły konfiguracji sprzętu szyfrującego połączonego z systemem klienta.

Tabela 873. Właściwości obiektu ConnectionFactory (kontynuacja)	
Nazwa właściwości	**Opis**
XMSC_WMQ_SSL_FIPS_REQUIRED	Wartość tej właściwości określa, czy aplikacja może lub nie może używać zestawów algorytmów szyfrowania niezgodnych ze standardem FIPS. Jeśli ta właściwość zostanie ustawiona na wartość true, w przypadku połączeń klient-serwer będzie można używać tylko algorytmów zgodnych ze standardem FIPS.
XMSC_WMQ_SSL_KEY_REPOSITORY	Położenie pliku bazy danych kluczy, w którym przechowywane są klucze i certyfikaty.
XMSC_WMQ_SSL_KEY_RESETCOUNT	Wartość KeyResetCount reprezentuje całkowitą liczbę nieszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed ponownym wynegocjowaniem klucza tajnego.
XMSC_WMQ_SSL_PEER_NAME	Nazwa węzła sieci używana na potrzeby połączenia SSL z menedżerem kolejek.
XMSC_WMQ_SYNCPOINT_ALL_GETS	Określa, czy wszystkie komunikaty muszą być pobierane z kolejek w ramach sterowania punktem synchronizacji.
“XMSC_WMQ_TARGET_CLIENT” na stronie 2152	
XMSC_WMQ_TEMP_Q_PREFIX	Przedrostek używany do tworzenia nazwy kolejki dynamicznej IBM MQ , która jest tworzona podczas tworzenia przez aplikację kolejki tymczasowej XMS .
XMSC_WMQ_TEMP_TOPIC_PREFIX	Podczas tworzenia tematów tymczasowych produkt XMS generuje łańcuch tematu w postaci "TEMP/TEMPTOPICPREFIX/unique_id" lub, jeśli ta właściwość zawiera wartość domyślną, generowany jest łańcuch "TEMP/unique_id". Podanie niepustej wartości umożliwia definiowanie konkretnych kolejek modelowych na potrzeby tworzenia zarządzanych kolejek dla subskrybentów tymczasowych tematów utworzonych w ramach tego połączenia.
XMSC_WMQ_TEMPORARY_MODEL	Nazwa kolejki modelowej IBM MQ , na podstawie której tworzona jest kolejka dynamiczna, gdy aplikacja tworzy kolejkę tymczasową XMS .
XMSC_WPM_BUS_NAME	W przypadku fabryki połączeń: nazwa magistrali integracji usług, z którą aplikacja nawiązuje połączenie. W przypadku miejsca docelowego: nazwa magistrali integracji usług, w której znajduje się miejsce docelowe.
XMSC_WPM_CONNECTION_BLISKOŚĆ	Ustawienie bliskości połączeń na potrzeby nawiązywania połączenia.
XMSC_WPM_DUR_SUB_HOME	Nazwa mechanizmu przesyłania komunikatów zarządzającego wszystkimi trwałymi subskrypcjami połączeń lub miejsc docelowych.
XMSC_WPM_LOCAL_ADDRESS	W przypadku połączenia z magistralą integracji usług: ta właściwość określa używaną wartość interfejsu sieci lokalnej albo portu lokalnego (lub zakresu portów lokalnych) bądź obie te wartości.

Tabela 873. Właściwości obiektu ConnectionFactory (kontynuacja)	
Nazwa właściwości	**Opis**
XMSC_WPM_NON_PERSISTENT_MAP	Poziom niezawodności komunikatów nietrwałych wysyłanych za pośrednictwem połączenia.
XMSC_WPM_PERSISTENT_MAP	Poziom niezawodności komunikatów trwałych wysyłanych za pośrednictwem połączenia.
PUNKTY KOŃCOWE DOSTAWCY XMSC_WPM_PROVIDER_ENDPOINTS	Jeden adres punktu końcowego serwera startowego lub sekwencja wielu adresów.
XMSC_WPM_TARGET_GROUP	Nazwa grupy docelowej mechanizmów przesyłania komunikatów.
XMSC_WPM_TARGET_IMPORTANCE	Znaczenie grupy docelowej mechanizmów przesyłania komunikatów.
XMSC_WPM_TARGET_TRANSPORT_CHAIN	Nazwa łańcucha transportowego danych przychodzących, który musi być używany przez aplikację do nawiązywania połączenia z mechanizmem przesyłania komunikatów.
XMSC_WPM_TARGET_TYPE	Typ grupy docelowej mechanizmów przesyłania komunikatów.
XMSC_WPM_TEMP_Q_PREFIX	Przedrostek używany do tworzenia nazwy kolejki tymczasowej, która jest tworzona w magistrali integracji usług podczas tworzenia przez aplikację kolejki tymczasowej XMS .
XMSC_WPM_TEMP_TOPIC_PREFIX	Przedrostek używany do generowania nazwy tematu tymczasowego tworzonego przez aplikację.

Właściwości danych ConnectionMeta

Przegląd właściwości obiektu danych ConnectionMetaz odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 874. Właściwości danych ConnectionMeta	
Nazwa właściwości	**Opis**
XMSC_JMS_MAJOR_VERSION (XMSC_JMS_MAJOR_VERSION)	Numer wersji głównej specyfikacji JMS , na której opiera się produkt XMS . Ta właściwość jest tylko do odczytu.
XMSC_JMS_MINOR_VERSION	Numer wersji podrzędnej specyfikacji JMS , na której opiera się produkt XMS . Ta właściwość jest tylko do odczytu.
XMSC_JMS_VERSION (XMSC_JMS_VERSION)	Identyfikator wersji specyfikacji JMS , na której opiera się produkt XMS . Ta właściwość jest tylko do odczytu.
XMSC_MAJOR_VERSION (XMSC_MAJOR_VERSION)	Numer wersji klienta XMS . Ta właściwość jest tylko do odczytu.
XMSC_MINOR_VERSION (XMSC_MINOR_VERSION)	Numer wersji klienta XMS . Ta właściwość jest tylko do odczytu.
NAZWA_DOSTAWCY XMSC_NAME	Dostawca klienta XMS . Ta właściwość jest tylko do odczytu.
WERSJA_XMSC_VERSION	Identyfikator wersjiXMSklienta. Ta właściwość jest tylko do odczytu.

Właściwości miejsca docelowego

Przegląd właściwości obiektu docelowego wraz z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 875. Właściwości miejsca docelowego

Nazwa właściwości	Opis
TRYB_XMSC_DELIVERY_MODE	Tryb dostarczania komunikatów wysyłanych do miejsca docelowego.
XMSC_PRIORITY	Priorytet komunikatów wysyłanych do miejsca docelowego.
XMSC_RTT_MULTICAST (XMSC_RTT_MULTICAST)	Ustawienie rozsyłania grupowego dla fabryki połączeń lub miejsca docelowego.
XMSC_TIME_TO_LIVE	Czas życia komunikatów wysyłanych do miejsca docelowego.
XMSC_WMQ_BROKER_VERSION	Typ brokera używany na potrzeby połączenia lub miejsca docelowego.
XMSC_WMQ_CCSID	Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, w którym znajdują się łańcuchy danych znakowych w treści komunikatu, gdy klient XMS przekazuje komunikat do miejsca docelowego.
XMSC_WMQ_DUR_SUBQ	Nazwa kolejki trwałego subskrybenta, który odbiera komunikaty z miejsca docelowego. **Uwaga:** Ta właściwość może być używana z wersją 2.0 produktu IBM Message Service Client for .NET , ale nie ma wpływu na aplikację połączoną z menedżerem kolejek produktu IBM WebSphere MQ 7.0 , chyba że właściwość XMSC_WMQ_PROVIDER_VERSION fabryki połączeń jest ustawiona na numer wersji mniejszy niż 7.
XMSC_WMQ_ENCODING	Sposób reprezentowania danych liczbowych w treści komunikatu, gdy klient XMS przekazuje komunikat do miejsca docelowego.
XMSC_WMQ_FAIL_IF_QUIESCE	Określa, czy wywołania pewnych metod nie powiodą się, jeśli menedżer kolejek, z którym aplikacja nawiązała połączenie, będzie w stanie wyciszania.
“XMSC_WMQ_MESSAGE_BODY” na stronie 2134	Ta właściwość określa, czy aplikacja XMS przetwarza MQRFH2 komunikatu IBM MQ jako część ładunku komunikatu (czyli jako część treści komunikatu).
“XMSC_WMQ_MQMD_MESSAGE_CONTEXT” na stronie 2135	Określa poziom kontekstu komunikatu, który ma zostać ustawiony przez aplikację XMS . Aby ta właściwość miała zastosowanie, aplikacja musi być uruchamiana z odpowiednimi uprawnieniami dotyczącymi kontekstu.
“XMSC_WMQ_MQMD_READ_ENABLED” na stronie 2136	Ta właściwość określa, czy aplikacja XMS może wyodrębniać wartości pól MQMD, czy nie.
“XMSC_WMQ_MQMD_WRITE_ENABLED” na stronie 2136	Ta właściwość określa, czy aplikacja XMS może ustawiać wartości pól MQMD.

Tabela 875. Właściwości miejsca docelowego (kontynuacja)	
Nazwa właściwości	**Opis**
“XMSC_WMQ_READ_AHEAD_ALLOWED” na stronie 2137	Ta właściwość określa, czy konsumenty komunikatów i przeglądarki kolejek mogą używać operacji odczytu z wyprzedzeniem do pobierania nietrwałych komunikatów nietransakcyjnych z tego miejsca docelowego do buforu wewnętrznego przed ich odebraniem.
“XMSC_WMQ_READ_AHEAD_CLOSE_POLICY” na stronie 2137	W przypadku komunikatów dostarczanych do asynchronicznego procesu nasłuchującego komunikatów ta właściwość określa, co stanie się z komunikatami w wewnętrznym buforze odczytu z wyprzedzeniem, gdy konsument komunikatów zostanie zamknięty.
“XMSC_WMQ_RECEIVE_CCSID” na stronie 2142	Właściwość miejsca docelowego, która ustawia docelowy identyfikator CCSID dla konwersji komunikatów menedżera kolejek. Ta wartość jest ignorowana, chyba że właściwość XMSC_WMQ_RECEIVE_CONVERSION jest ustawiona na wartość WMQ_RECEIVE_CONVERSION_QMGR.
“XMSC_WMQ_RECEIVE_CONVERSION” na stronie 2142	Właściwość miejsca docelowego, która określa, czy konwersja danych będzie wykonywana przez menedżer kolejek.
XMSC_WMQ_TARGET_CLIENT	Określa, czy komunikaty wysyłane do miejsca docelowego mają zawierać nagłówek MQRFH2.
XMSC_WMQ_TEMP_TOPIC_PREFIX	Podczas tworzenia tematów tymczasowych produkt XMS generuje łańcuch tematu w postaci "TEMP/TEMPTOPICPREFIX/unique_id" lub, jeśli ta właściwość zawiera wartość domyślną, generowany jest łańcuch "TEMP/unique_id". Podanie niepustej wartości umożliwia definiowanie konkretnych kolejek modelowych na potrzeby tworzenia zarządzanych kolejek dla subskrybentów tymczasowych tematów utworzonych w ramach tego połączenia.
XMSC_WPM_BUS_NAME	W przypadku fabryki połączeń: nazwa magistrali integracji usług, z którą aplikacja nawiązuje połączenie. W przypadku miejsca docelowego: nazwa magistrali integracji usług, w której znajduje się miejsce docelowe.
XMSC_WPM_TOPIC_SPACE	Nazwa obszaru tematu zawierającego dany temat.

Właściwości kontekstu InitialContext

Przegląd właściwości obiektu InitialContext z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 876. Właściwości kontekstu InitialContext	
Nazwa właściwości	**Opis**
XMSC_IC_PROVIDER_URL (XMSC_PROVIDER_URL)	Umożliwia wskazanie katalogu nazw JNDI. Dzięki niej usługa nazw COS nie musi znajdować się na tym samym serwerze co usługa WWW.
XMSC_IC_SECURITY_AUTHENTICATION	Na podstawie interfejsu kontekstu Java SECURITY_AUTHENTICATION. Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.

Tabela 876. Właściwości kontekstu InitialContext (kontynuacja)	
Nazwa właściwości	**Opis**
XMSC_IC_SECURITY_CREDENTIALS	W oparciu o Java Context interface SECURITY_CREDENTIALS. Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.
XMSC_IC_SECURITY_PRINCIPAL	W oparciu o interfejs kontekstu Java SECURITY_PRINCIPAL. Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.
XMSC_IC_SECURITY_PROTOCOL	Na podstawie parametru Java Context interface SECURITY_PROTOCOL Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.
Adres XMSC_IC_URL	W przypadku kontekstów LDAP i FileSystem: adres repozytorium zawierającego obiekty administrowane. W przypadku kontekstów nazw COS: adres usługi WWW, która wyszukuje obiekty w katalogu.

Właściwości komunikatu

Przegląd właściwości obiektu Message z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 877. Właściwości komunikatu	
Nazwa właściwości	**Opis**
JMS_IBM_character TER_SET (zestaw znaków JMS_IBM_character)	Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, w którym znajdują się łańcuchy danych znakowych w treści komunikatu, gdy klient XMS przekazuje komunikat do zamierzonego miejsca docelowego. W przypadku klienta XMS wartością tej właściwości jest liczba, która jest odwzorowywana na identyfikator CCSID. Ta właściwość jest jednak oparta na właściwości JMS, dlatego przyjmuje wartość typu String, która jest odwzorowywana na zestaw znaków języka Java reprezentujący ten liczbowy identyfikator CCSID.
JMS_IBM_Encoding	Sposób reprezentowania danych liczbowych w treści komunikatu, gdy klient XMS przekazuje komunikat do zamierzonego miejsca docelowego.
JMS_IBM_EXCEPTIONMESSAGE	Tekst opisujący przyczynę, z powodu której komunikat został wysłany do miejsca docelowego wyjątków. Ta właściwość jest tylko do odczytu.
JMS_IBM_EXCEPTIONPROBLEMDESTINATION	Nazwa miejsca docelowego, w którym znajdował się komunikat, zanim został wysłany do miejsca docelowego wyjątków.
JMS_IBM_EXCEPTIONREASON	Kod przyczyny wskazujący powód wysłania komunikatu do miejsca docelowego wyjątków.
JMS_IBM_EXCEPTIONTIMESTAMP	Czas wysłania komunikatu do miejsca docelowego wyjątków.
JMS_IBM_FEEDBACK	Kod wskazujący rodzaj komunikatu raportu.
JMS_IBM_FORMAT	Rodzaj danych aplikacji w komunikacie.

Tabela 877. Właściwości komunikatu (kontynuacja)	
Nazwa właściwości	**Opis**
JMS_IBM_LAST_MSG_IN_GROUP	Wskazuje, czy komunikat jest ostatnim komunikatem w grupie komunikatów.
JMS_IBM_MSGTYPE	Typ komunikatu.
JMS_IBM_PUTAPPLTYPE	Typ aplikacji, która wysłała komunikat.
JMS_IBM_PUTDATE	Data wysłania komunikatu.
JMS_IBM_PUTTIME (Czas wywołania JMS IBM_PUTTIME)	Godzina wysłania komunikatu.
JMS_IBM_REPORT_COA	Wysyła żądanie przesyłania komunikatów raportów 'potwierdzenie odebrania' i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.
JMS_IBM_REPORT_COD	Wysyła żądanie przesyłania komunikatów raportów 'potwierdzenie dostarczenia' i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.
JMS_IBM_REPORT_DISCARD_MSG	Wysyła żądanie, aby komunikat był usuwany, jeśli nie może zostać dostarczony do wybranego miejsca docelowego.
JMS_IBM_REPORT_EXCEPTION (wyjątek JMS_IBM_REPORT_EXCEPTION)	Wysyła żądanie przesyłania komunikatów raportów o wyjątkach i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.
JMS_IBM_REPORT_EXPIRATION	Wysyła żądanie przesyłania komunikatów raportów o utracie ważności i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.
JMS_IBM_REPORT_NAN,	Wysyła żądanie przesyłania komunikatów raportów o powiadomieniach dotyczących działań negatywnych.
JMS_IBM_REPORT_PAN,	Wysyła żądanie przesyłania komunikatów raportów o powiadomieniach dotyczących działań pozytywnych.
JMS_IBM_REPORT_PASS_CORREL_ID	Wysyła żądanie, aby identyfikator korelacji dowolnego komunikatu odpowiedzi lub raportu był taki sam jak identyfikator korelacji oryginalnego komunikatu.
JMS_IBM_REPORT_PASS_MSG_ID	Wysyła żądanie, aby identyfikator dowolnego komunikatu odpowiedzi lub raportu był taki sam jak identyfikator oryginalnego komunikatu.
JMS_IBM_RETAIN (JMS_IBM_RETAIN)	Jeśli ta właściwość zostanie ustawiona, menedżer kolejek będzie traktował komunikat jako zachowaną publikację.
JMS_IBM_SYSTEM_MESSAGEID	Identyfikator, który identyfikuje komunikat w jednoznaczny sposób w obrębie magistrali integracji usług. Ta właściwość jest tylko do odczytu.
JMSX_APPID	Nazwa aplikacji, która wysłała komunikat.
LICZNIK_OGRANICZENIA_JMS_XX_ENCODE_CASE_CAPS_LOCK_OFF	Liczba prób dostarczenia komunikatu.

Tabela 877. Właściwości komunikatu (kontynuacja)	
Nazwa właściwości	**Opis**
ID_GRUPY_JMS	Identyfikator grupy komunikatów, do której należy komunikat.
JMSX_GROUPSEQ,	Numer kolejny komunikatu w obrębie grupy komunikatów.
JMSX_USERID (identyfikator JMSX_USERID)	Identyfikator użytkownika powiązany z aplikacją, która wysłała komunikat.

Właściwości JMS_IBM_MQMD*

Program IBM Message Service Client for .NET umożliwia aplikacjom klienckim odczytywanie i zapisywanie pól MQMD za pomocą interfejsów API. Umożliwia również dostęp do danych komunikatów produktu MQ . Domyślnie dostęp do struktury MQMD jest wyłączony i musi zostać jawnie włączony przez aplikację przy użyciu właściwości miejsca docelowego XMSC_WMQ_MQMD_WRITE_ENABLED i XMSC_WMQ_MQMD_READ_ENABLED. Te dwie właściwości są od siebie niezależne.

Wszystkie pola MQMD z wyjątkiem StrucId i Version są prezentowane jako dodatkowe właściwości obiektu Message i mają przedrostek JMS_IBM_MQMD.

Właściwości JMS_IBM_MQMD* mają wyższy priorytet niż inne właściwości, takie jak JMS_IBM* opisane w poprzedniej tabeli.

Wysyłanie komunikatów

Reprezentowane są wszystkie pola MQMD z wyjątkiem StrucId i Version. Te właściwości odnoszą się tylko do pól MQMD. W przypadku, gdy właściwość występuje zarówno w deskryptorze MQMD, jak i w nagłówku MQRFH2 , wersja w nagłówku MQRFH2 nie jest ustawiana ani wyodrębniana. Można ustawić dowolną z tych właściwości, z wyjątkiem właściwości JMS_IBM_MQMD_BackoutCount. Każda wartość ustawiona dla JMS_IBM_MQMD_BackoutCount jest ignorowana.

Jeśli właściwość ma maksymalną długość i zostanie podana wartość, która jest zbyt długa, wartość zostanie obcięta.

W przypadku niektórych właściwości należy również ustawić właściwość XMSC_WMQ_MQMD_MESSAGE_CONTEXT w obiekcie docelowym. Aby ta właściwość miała zastosowanie, aplikacja musi być uruchamiana z odpowiednimi uprawnieniami dotyczącymi kontekstu. Jeśli parametr XMSC_WMQ_MQMD_MESSAGE_CONTEXT nie zostanie ustawiony na odpowiednią wartość, wartość właściwości zostanie zignorowana. Jeśli właściwość XMSC_WMQ_MQMD_MESSAGE_CONTEXT zostanie ustawiona na odpowiednią wartość, ale użytkownik nie ma wystarczających uprawnień kontekstowych dla menedżera kolejek, zostanie zgłoszony wyjątek. Poniżej przedstawiono właściwości wymagające konkretnych wartości właściwości XMSC_WMQ_MQMD_MESSAGE_CONTEXT.

Następujące właściwości wymagają ustawienia właściwości XMSC_WMQ_MQMD_MESSAGE_CONTEXT na wartość XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT lub XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_UserIdentifier
- JMS_IBM_MQMD_AccountingToken
- Dane JMS_IBM_MQMD_ApplIdentity

Następujące właściwości wymagają ustawienia właściwości XMSC_WMQ_MQMD_MESSAGE_CONTEXT na wartość XMSC_WMQ_MDCTX_SET_ALL_CONTEXT:

- JMS_IBM_MQMD_PutApplTyp
- JMS_IBM_MQMD_PutApplNazwa
- JMS_IBM_MQMD_PutDate
- JMS_IBM_MQMD_PutTime
- Dane JMS_IBM_MQMD_ApplOrigin

Odbieranie komunikatów

Wszystkie te właściwości są dostępne w odebranym komunikacie, jeśli właściwość XMSC_WMQ_MQMD_READ_ENABLED jest ustawiona na wartość true, niezależnie od rzeczywistych właściwości ustawionych przez aplikację generującą. Aplikacja nie może modyfikować właściwości odebranego komunikatu, chyba że wszystkie właściwości zostaną najpierw wyczyszczone zgodnie ze specyfikacją JMS. Odebrany komunikat można przekazać bez modyfikowania właściwości.

Uwaga: Jeśli aplikacja odbiera komunikat z miejsca docelowego z właściwością XMSC_WMQ_MQMD_READ_ENABLED ustawioną na wartość true i przekazuje go do miejsca docelowego z właściwością XMSC_WMQ_MQMD_WRITE_ENABLED ustawioną na wartość true, powoduje to skopiowanie wszystkich wartości pól MQMD odebranego komunikatu do przekazywanego komunikatu. Tabela właściwości

Tabela 878. Właściwości obiektu Message reprezentującego pola MQMD

Właściwość	**Opis**	**Typ**
RAPORT JMS_IBM_MQMD_REPORT	Opcje dla komunikatów raportu	System.Int32
JMS_IBM_MQMD_MSGTYPE	Typ komunikatu	System.Int32
JMS_IBM_MQMD_WAŻNOŚĆ	czas życia komunikatu	System.Int32
JMS_IBM_MQMD_FEEDBACK	Informacja zwrotna lub kod przyczyny	System.Int32
JMS_IBM_MQMD_ENCODING	Kodowanie liczbowe danych komunikatu	System.Int32
JMS_IBM_MQMD_CODEDCHARSETID,	Identyfikator zestawu znaków danych komunikatu	System.Int32
FORMAT JMS_IBM_MQMD_FORMAT	Nazwa formatu danych komunikatu	System.String
JMS_IBM_MQMD_PRIORITY, **Uwaga:** Jeśli do parametru JMS_IBM_MQMD_PRIORITY zostanie przypisana wartość spoza zakresu od 0 do 9, wartość ta będzie niezgodna ze specyfikacją JMS.	Priorytet komunikatu	System.Int32
JMS_IBM_MQMD_PERSISTENCE	Trwałość komunikatu	System.Int32
JMS_IBM_MQMD_MSGID **Uwaga:** Specyfikacja JMS określa, że identyfikator komunikatu musi być ustawiony przez dostawcę JMS i musi być unikalny lub mieć wartość NULL. Jeśli wartość zostanie przypisana do JMS_IBM_MQMD_MSGID, zostanie ona skopiowana do JMSMessageID. Dlatego nie jest ona ustawiana przez dostawcę JMS i może nie być unikalna: ta wartość narusza specyfikację JMS.	Identyfikator komunikatu	Tablica bajtów **Uwaga:** Użycie właściwości tablicy bajtów w komunikacie narusza specyfikację JMS.

Tabela 878. Właściwości obiektu Message reprezentującego pola MQMD (kontynuacja)		
Właściwość	**Opis**	**Typ**
JMS_IBM_MQMD_CORRELID, **Uwaga:** Jeśli do JMS_IBM_MQMD_CORRELID zostanie przypisana wartość rozpoczynająca się od łańcucha ID:, spowoduje to naruszenie specyfikacji JMS.	Identyfikator korelacji	Tablica bajtów **Uwaga:** Użycie właściwości tablicy bajtów w komunikacie narusza specyfikację JMS.
JMS_IBM_MQMD_BACKOUTCOUNT	Liczba wycofanych	System.Int32
JMS_IBM_MQMD_REPLYTOQ	Nazwa kolejki odpowiedzi	System.String
JMS_IBM_MQMD_REPLYTOQMGR	Nazwa menedżera kolejek odpowiedzi	System.String
JMS_IBM_MQMD_USERIDENTIFIER	Identyfikator użytkownika	System.String
JMS_IBM_MQMD_ACCOUNTINGTOKEN,	Token rozliczania	Tablica bajtów **Uwaga:** Użycie właściwości tablicy bajtów w komunikacie narusza specyfikację JMS.
JMS_IBM_MQMD_APPLIDENTITYDATA	Dane aplikacji odnoszące się do tożsamości	System.String
JMS_IBM_MQMD_PUTAPPLTYPE	Typ aplikacji, która umieściła komunikat	System.Int32
JMS_IBM_MQMD_PUTAPPLNAME	Nazwa aplikacji, która umieściła komunikat	System.String
JMS_IBM_MQMD_PUTDATE	Data umieszczenia komunikatu	System.String
JMS_IBM_MQMD_PUTTIME	Czas umieszczenia komunikatu	System.String
JMS_IBM_MQMD_APPLORIGINDATA	Dane aplikacji odnoszące się do pochodzenia	System.String
JMS_IBM_MQMD_GROUPID	Identyfikator grupy	Tablica bajtów **Uwaga:** Użycie właściwości tablicy bajtów w komunikacie narusza specyfikację JMS.
JMS_IBM_MQMD_MSGSEQNUMBER	Numer kolejny komunikatu lokalnego w grupie	System.Int32
JMS_IBM_MQMD_OFFSET	Przesunięcie danych w komunikacie fizycznym od początku komunikatu logicznego	System.Int32
JMS_IBM_MQMD_MSGFLAGS	Flagi komunikatów	System.Int32
JMS_IBM_MQMD_ORIGINALLENGTH	Długość oryginalnego komunikatu	System.Int32

Więcej informacji na ten temat zawiera sekcja MQMD .

Przykłady

W tym przykładzie komunikat jest umieszczany w kolejce lub temacie za pomocą programu MQMD.UserIdentifier ustawiony na "JoeBloggs".

```
// Create a ConnectionFactory, connection, session, producer, message
// ...

// Create a destination
// ...

// Enable MQMD write
dest.setBooleanProperty(XMSC_WMQ_MQMD_WRITE_ENABLED,
      XMSC_WMQ_MQMD_WRITE_ENABLED_YES);

// Optionally, set a message context if applicable for this MD field
dest.setIntProperty(XMSC_WMQ_MQMD_MESSAGE_CONTEXT,
  XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT);

// On the message, set property to provide custom UserId
msg.setStringProperty(JMS_IBM_MQMD_USERIDENTIFIER, "JoeBloggs");

// Send the message
// ...
```

Przed ustawieniem zmiennej JMS_IBM_MQMD_USERIDENTIFIER należy ustawić zmienną XMSC_WMQ_MQMD_MESSAGE_CONTEXT. Więcej informacji na temat używania właściwości XMSC_WMQ_MQMD_MESSAGE_CONTEXT zawiera sekcja Właściwości obiektu komunikatu.

Podobnie można wyodrębnić treść pól MQMD, ustawiając właściwość XMSC_WMQ_MQMD_READ_ENABLED na wartość true przed odebraniem komunikatu, a następnie przy użyciu metod pobierania komunikatu, takich jak właściwość getString. Wszystkie odebrane właściwości są tylko do odczytu.

W tym przykładzie wynikiem jest pole wartości zawierające wartość deskryptora MQMD produktu MQMD.ApplIdentityData komunikatu uzyskanego z kolejki lub tematu.

```
// Create a ConnectionFactory, connection, session, consumer
// ...

// Create a destination
// ...

// Enable MQMD read
dest.setBooleanProperty(XMSC_WMQ_MQMD_READ_ENABLED, XMSC_WMQ_MQMD_READ_ENABLED_YES);

// Receive a message
// ...

// Get required MQMD field value using a property
System.String value = rcvMsg.getStringProperty(JMS_IBM_MQMD_APPLIDENTITYDATA);
```

Właściwości elementu MessageConsumer

Przegląd właściwości obiektu MessageConsumer z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Tabela 879. Właściwości elementu MessageConsumer

Nazwa właściwości	Opis
XMSC_IS_SUBSCRIPTION_MULTICAST (XMSC_IS_SUBSCRIPTION_MULTICAST)	Wskazuje, czy komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport. Ta właściwość jest tylko do odczytu.

Tabela 879. Właściwości elementu MessageConsumer (kontynuacja)	
Nazwa właściwości	**Opis**
XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST (xmsc_is_subscription_reliable_multicast)	Wskazuje, czy komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport z niezawodną jakością usługi. Ta właściwość jest tylko do odczytu.

Patrz .Więcej szczegółów zawierają właściwości NET interfejsu IMessageConsumer .

Właściwości producenta MessageProducer

Przegląd właściwości obiektu MessageProducer z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Patrz .Więcej szczegółów zawierają właściwości NET IMessageProducer .

Właściwości sesji

Przegląd właściwości obiektu sesji z odsyłaczami do bardziej szczegółowych informacji uzupełniających.

Patrz .Aby uzyskać więcej szczegółów, należy wyświetlić właściwości NET interfejsu ISession .

Definicje właściwości

Ta sekcja zawiera definicję każdej właściwości obiektu.

Każda definicja właściwości zawiera następujące informacje:

- typ danych właściwości;
- Typy obiektów, które mają właściwość
- W przypadku właściwości Destination jest to nazwa, która może być używana w identyfikatorze URI.
- Bardziej szczegółowy opis właściwości
- Poprawne wartości właściwości
- Wartość domyślna właściwości.

Właściwości, których nazwy rozpoczynają się jednym z następujących przedrostków, mają zastosowanie tylko w przypadku określonego typu połączenia:

XMSC_RTT
Właściwości mają zastosowanie tylko w przypadku połączenia z brokerem w czasie rzeczywistym. Nazwy właściwości są definiowane jako stałe nazwane w pliku nagłówkowym `xmsc_rtt.h`.

XMSC_WMQ
Właściwości mają zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek produktu IBM MQ . Nazwy właściwości są definiowane jako stałe nazwane w pliku nagłówkowym `xmsc_wmq.h`.

XMSC_WPM,
Właściwości mają zastosowanie tylko wtedy, gdy aplikacja łączy się z magistralą integracji usług produktu WebSphere . Nazwy właściwości są definiowane jako stałe nazwane w pliku nagłówkowym `xmsc_wpm.h`.

O ile nie określono inaczej w ich definicjach, pozostałe właściwości są istotne dla wszystkich typów połączeń. Nazwy właściwości są definiowane jako stałe nazwane w pliku nagłówkowym `xmsc.h`. Właściwości, których nazwy rozpoczynają się od przedrostka JMSX, są JMS właściwościami zdefiniowanymi dla komunikatu, a właściwości, których nazwy rozpoczynają się od przedrostka JMS_IBM, są IBM właściwościami zdefiniowanymi dla komunikatu. Więcej informacji na temat właściwości komunikatów zawiera sekcja Właściwości komunikatu XMS.

O ile w swojej definicji nie określono inaczej, każda właściwość ma zastosowanie zarówno w domenach typu punkt z punktem, jak i w domenach publikowania subskrybowania.

Aplikacja może pobrać i ustawić wartość dowolnej właściwości, chyba że właściwość jest oznaczona jako tylko do odczytu.

JMS_IBM_CHARACTER_SET (zestaw znaków IMS)

Typ danych:
System.Int32

Właściwość:
Komunikat

Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, w którym znajdują się łańcuchy danych znakowych w treści komunikatu, gdy klient XMS przekazuje komunikat do zamierzonego miejsca docelowego. W przypadku klienta XMS wartością tej właściwości jest liczba, która jest odwzorowywana na identyfikator CCSID. Ta właściwość jest jednak oparta na właściwości JMS, dlatego przyjmuje wartość typu String, która jest odwzorowywana na zestaw znaków języka Java reprezentujący ten liczbowy identyfikator CCSID. Ta właściwość nadpisuje wszystkie identyfikatory CCSID określone dla miejsca docelowego przez właściwość XMSC_WMQ_CCSID .

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_Encoding

Typ danych:
System.Int32

Właściwość:
Komunikat

Sposób reprezentowania danych liczbowych w treści komunikatu, gdy klient XMS przekazuje komunikat do zamierzonego miejsca docelowego. Ta właściwość przesłania dowolne kodowanie określone dla miejsca docelowego przez właściwość XMSC_WMQ_ENCODING . Ta właściwość określa reprezentację binarnych liczb całkowitych, upakowanych liczb dziesiętnych i liczb zmiennopozycyjnych.

Poprawne wartości właściwości są takie same, jak wartości, które można podać w polu **Encoding** deskryptora komunikatu.

Aplikacja może użyć następujących stałych nazwanych w celu ustawienia właściwości:

Stała nazwana	Znaczenie
MQENC_INTEGER_NORMAL	Normalne kodowanie liczb całkowitych
MQENC_INTEGER_REVERSED,	Kodowanie odwrotnej liczby całkowitej
MQENC_DECIMAL_NORMAL	Normalne kodowanie upakowanych liczb dziesiętnych
MQENC_DECIMAL_REVERSED (odwrócone)	Odwrócone upakowane kodowanie dziesiętne
MQENC_FLOAT_IEEE_NORMAL	Normalne kodowanie zmiennopozycyjne IEEE
MQENC_FLOAT_IEEE_REVERSED	Odwrotne kodowanie zmiennopozycyjne IEEE
MQENC_FLOAT_S390	Kodowanie zmiennopozycyjne architektury z/OS
RODZIMA MQENC	Kodowanie na komputerze rodzimym

Aby utworzyć wartość właściwości, aplikacja może dodać trzy z następujących stałych:

- Stała, której nazwa rozpoczyna się od MQENC_INTEGER, służąca do określania reprezentacji binarnych liczb całkowitych.
- Stała, której nazwa rozpoczyna się od MQENC_DECIMAL, służąca do określania reprezentacji upakowanych dziesiętnych liczb całkowitych.
- Stała, której nazwa rozpoczyna się od MQENC_FLOAT, służąca do określania reprezentacji liczb zmiennopozycyjnych.

Alternatywnie aplikacja może ustawić tę właściwość na wartość MQENC_NATIVE, której wartość jest zależna od środowiska.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_EXCEPTIONMESSAGE

Typ danych:
Łańcuch

Właściwość:
Komunikat

Tekst opisujący przyczynę, z powodu której komunikat został wysłany do miejsca docelowego wyjątków. Ta właściwość jest tylko do odczytu.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z magistralą integracji usług i odbiera komunikat z miejsca docelowego wyjątków.

JMS_IBM_EXCEPTIONPROBLEMDESTINATION

Typ danych:
Łańcuch

Właściwość:
Komunikat

Nazwa miejsca docelowego, w którym znajdował się komunikat, zanim został wysłany do miejsca docelowego wyjątków.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z magistralą integracji usług i odbiera komunikat z miejsca docelowego wyjątków.

JMS_IBM_EXCEPTIONREASON

Typ danych:
System.Int32

Właściwość:
Komunikat

Kod przyczyny wskazujący powód wysłania komunikatu do miejsca docelowego wyjątków.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z magistralą integracji usług i odbiera komunikat z miejsca docelowego wyjątków.

JMS_IBM_EXCEPTIONTIMESTAMP

Typ danych:
System.Int64

Właściwość:
Komunikat

Czas wysłania komunikatu do miejsca docelowego wyjątków.

Czas jest wyrażony w milisekundach od godziny 00:00:00 GMT w dniu 1 stycznia 1970.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z magistralą integracji usług i odbiera komunikat z miejsca docelowego wyjątków.

JMS_IBM_FEEDBACK,

Typ danych:
System.Int32

Właściwość:
Komunikat

Kod wskazujący rodzaj komunikatu raportu.

Poprawne wartości właściwości to kody informacji zwrotnych i kody przyczyny, które można podać w polu **Feedback** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

JMS_IBM_FORMAT,

Typ danych:
Łańcuch

Właściwość:
Komunikat

Rodzaj danych aplikacji w komunikacie.

Poprawne wartości właściwości są takie same, jak wartości, które można podać w polu **Format** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_LAST_MSG_IN_GROUP

Typ danych:
System.Boolean

Właściwość:
Komunikat

Wskazuje, czy komunikat jest ostatnim komunikatem w grupie komunikatów.

Ustaw tę właściwość na wartość true, jeśli komunikat jest ostatnim komunikatem w grupie komunikatów. W przeciwnym razie ustaw wartość false dla tej właściwości lub nie ustawiaj jej. Domyślnie właściwość nie jest ustawiona.

Wartość true odpowiada fladze statusu MQMF_LAST_MSG_IN_GROUP, którą można określić w polu **MsgFlags** deskryptora komunikatu.

Ta właściwość jest ignorowana w domenie publikowania/subskrypcji i nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_MSGTYPE

Typ danych:
System.Int32

Właściwość:
Komunikat

Typ komunikatu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQMT_DATAGRAM,	Komunikat nie wymaga odpowiedzi.
MQMT_ŻĄDANIE	Komunikat wymaga odpowiedzi.
MQMT_REPLY	Komunikat jest komunikatem odpowiedzi.
MQMT_RAPORT	Komunikat jest raportem.

Te wartości odpowiadają typom komunikatów, które można określić w polu **MsgType** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_PUTAPPLTYPE

Typ danych:
System.Int32

Właściwość:
Komunikat

Typ aplikacji, która wysłała komunikat.

Poprawne wartości właściwości to typy aplikacji, które można określić w polu **PutApplType** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_PUTDATE,

Typ danych:
Łańcuch

Właściwość:
Komunikat

Data wysłania komunikatu.

Poprawne wartości właściwości są takie same, jak wartości, które można podać w polu **PutDate** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

CZAS JMS_IBM_PUTTIME

Typ danych:
Łańcuch

Właściwość:
Komunikat

Godzina wysłania komunikatu.

Poprawne wartości właściwości są takie same, jak wartości, które można podać w polu **PutTime** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMS_IBM_REPORT_COA

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie przesyłania komunikatów raportów 'potwierdzenie odebrania' i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQRO_COA	Żądanie "potwierdzenia przy odbiorze" komunikatów raportu, bez danych aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.
MQRO_KOA_WITH_DATA	Żądanie "potwierdzenia przy odbiorze" zgłasza komunikaty z pierwszymi 100 bajtami danych aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.
MQRO_KOA_WITH_FULL_DATA	Zażądaj komunikatów raportu potwierdzenia przy odbiorze, ze wszystkimi danymi aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.

Te wartości odpowiadają opcjom raportu, które można określić w polu **Report** deskryptora komunikatu. Więcej informacji na temat tych opcji zawiera sekcja Raport (MQLONG).

Domyślnie właściwość nie jest ustawiona.

JMS_IBM_REPORT_COD

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie przesyłania komunikatów raportów 'potwierdzenie dostarczenia' i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQRO_COD	Zażądaj komunikatów raportu potwierdzenia dostarczenia bez danych aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.
MQRO_KOD_WITH_DATA	Żądanie "potwierdzenia przy dostarczeniu" komunikatów raportu, z pierwszymi 100 bajtami danych aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.
MQRO_KOD_WITH_FULL_DATA	Zażądaj komunikatów raportu 'Potwierdź przy dostarczeniu' ze wszystkimi danymi aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.

Te wartości odpowiadają opcjom raportu, które można określić w polu **Report** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

JMS_IBM_REPORT_DISCARD_MSG

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie, aby komunikat był usuwany, jeśli nie może zostać dostarczony do wybranego miejsca docelowego.

Ustaw tę właściwość na wartość MQRO_DISCARD_MSG, aby zażądać usunięcia komunikatu, jeśli nie można go dostarczyć do zamierzonego miejsca docelowego. Jeśli komunikat ma zostać umieszczony

w kolejce niedostarczonych komunikatów lub wysłany do miejsca docelowego wyjątków, nie należy ustawiać tej właściwości. Domyślnie właściwość nie jest ustawiona.

Wartość MQRO_DISCARD_MSG odpowiada opcji raportu, którą można podać w polu **Report** deskryptora komunikatu.

JMS_IBM_REPORT_EXCEPTION,

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie przesyłania komunikatów raportów o wyjątkach i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQRO_EXCEPTION,	Żądaj komunikatów raportu o wyjątkach, bez danych aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.
MQRO_EXCEPTION_WITH_DATA	Żądaj komunikatów raportu o wyjątkach, z pierwszymi 100 bajtami danych aplikacji z oryginalnego komunikatu dołączonego do komunikatu raportu.
MQRO_EXCEPTION_WITH_FULL_DATA	Żądaj komunikatów raportu o wyjątkach, ze wszystkimi danymi aplikacji z oryginalnego komunikatu dołączonymi do komunikatu raportu.

Te wartości odpowiadają opcjom raportu, które można określić w polu **Report** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

JMS_IBM_REPORT_EXPIRATION

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie przesyłania komunikatów raportów o utracie ważności i określa, ile danych aplikacji z oryginalnego komunikatu musi zostać dołączonych do komunikatu raportu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQRO_UTR_WAŻN.	Komunikaty raportu o utracie ważności żądania, bez danych aplikacji z oryginalnego komunikatu zawartego w komunikacie raportu.
MQRO_EXPIRATION_WITH_DATA	Komunikaty raportu o utracie ważności żądania z pierwszymi 100 bajtami danych aplikacji z oryginalnego komunikatu dołączonymi do komunikatu raportu.

Poprawna wartość	Znaczenie
MQRO_EXPIRATION_WITH_FULL_DATA	Komunikaty raportu o utracie ważności żądania z wszystkimi danymi aplikacji z oryginalnego komunikatu dołączonymi do komunikatu raportu.

Te wartości odpowiadają opcjom raportu, które można określić w polu **Report** deskryptora komunikatu.

Domyślnie właściwość nie jest ustawiona.

JMS_IBM_REPORT_NAN

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie przesyłania komunikatów raportów o powiadomieniach dotyczących działań negatywnych.

Ustaw tę właściwość na wartość MQRO_NAN, aby zażądać komunikatów raportu powiadomienia o negatywnym działaniu. Jeśli nie są wymagane komunikaty raportu powiadomienia o działaniu negatywnym, nie należy ustawiać tej właściwości. Domyślnie właściwość nie jest ustawiona.

Wartość MQRO_NAN odpowiada opcji raportu, którą można podać w polu **Report** deskryptora komunikatu.

JMS_IBM_REPORT_PAN

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie przesyłania komunikatów raportów o powiadomieniach dotyczących działań pozytywnych.

Ustaw tę właściwość na wartość MQRO_PAN, aby żądać komunikatów raportu powiadomień o działaniach pozytywnych. Jeśli nie są wymagane komunikaty raportu powiadomienia o działaniu pozytywnym, nie należy ustawiać tej właściwości. Domyślnie właściwość nie jest ustawiona.

Wartość MQRO_PAN odpowiada opcji raportu, którą można podać w polu **Report** deskryptora komunikatu.

JMS_IBM_REPORT_PASS_CORREL_ID

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie, aby identyfikator korelacji dowolnego komunikatu odpowiedzi lub raportu był taki sam jak identyfikator korelacji oryginalnego komunikatu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQRO_PASS_CORREL_ID (Identyfikator KORELACJI MQ)	Wysyła żądanie, aby identyfikator korelacji dowolnego komunikatu odpowiedzi lub raportu był taki sam jak identyfikator korelacji oryginalnego komunikatu.

Poprawna wartość	Znaczenie
MQRO_COPY_MSG_ID_TO_CORREL_ID	Zażądaj, aby identyfikator korelacji dowolnego komunikatu raportu lub odpowiedzi był taki sam jak identyfikator oryginalnego komunikatu.

Te wartości odpowiadają opcjom raportu, które można określić w polu **Report** deskryptora komunikatu.

Wartością domyślną tej właściwości jest MQRO_COPY_MSG_ID_TO_CORREL_ID.

JMS_IBM_REPORT_PASS_MSG_ID

Typ danych:
System.Int32

Właściwość:
Komunikat

Wysyła żądanie, aby identyfikator dowolnego komunikatu odpowiedzi lub raportu był taki sam jak identyfikator oryginalnego komunikatu.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
MQRO_PASS_MSG_ID	Wysyła żądanie, aby identyfikator dowolnego komunikatu odpowiedzi lub raportu był taki sam jak identyfikator oryginalnego komunikatu.
MQRO_NEW_MSG_ID	Żądanie wygenerowania nowego identyfikatora komunikatu dla każdego komunikatu raportu lub komunikatu odpowiedzi.

Te wartości odpowiadają opcjom raportu, które można określić w polu Raport deskryptora komunikatu.

Wartością domyślną tej właściwości jest MQRO_NEW_MSG_ID.

JMS_IBM_RETAIN,

Typ danych:
System.Int32

Właściwość:
Komunikat

Jeśli ta właściwość zostanie ustawiona, menedżer kolejek będzie traktował komunikat jako zachowaną publikację. Kiedy subskrybent odbiera komunikaty z tematów, może odbierać dodatkowe komunikaty natychmiast po zasubskrybowaniu, poza komunikatami odebranymi w poprzednich wersjach. Te komunikaty to opcjonalne zachowane publikacje dla subskrybowanych tematów. Dla każdego tematu zgodnego z subskrypcją, jeśli istnieje zachowana publikacja, jest ona udostępniana do dostarczenia do subskrybującego konsumenta komunikatów.

Jedyną poprawną wartością tej właściwości jest RETAIN_PUBLICATION. Domyślnie ta właściwość nie jest ustawiona.

Uwaga: Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji

JMS_IBM_SYSTEM_MESSAGEID

Typ danych:
Łańcuch

Właściwość:
Komunikat

Identyfikator, który identyfikuje komunikat w jednoznaczny sposób w obrębie magistrali integracji usług. Ta właściwość jest tylko do odczytu.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z magistralą integracji usług.

JMSX_APPID

Typ danych:
Łańcuch

Właściwość:
Komunikat

Nazwa aplikacji, która wysłała komunikat.

Ta właściwość jest zdefiniowaną właściwością JMS o nazwie JMS name JMSXAppID. Więcej informacji na temat tej właściwości zawiera dokumentacja *Java Message Service Specification, wersja 1.1*.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie jest poprawna w przypadku połączenia z brokerem w czasie rzeczywistym.

JMSX_DELIVERY_COUNT (licznik ogranicznik_JMS)

Typ danych:
System.Int32

Właściwość:
Komunikat

Liczba prób dostarczenia komunikatu.

Ta właściwość jest właściwością zdefiniowaną przez JMS o nazwie JMS name JMSXDeliveryCount. Więcej informacji na temat tej właściwości zawiera specyfikacja *Java Message Service , wersja 1.1*.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie jest poprawna w przypadku połączenia z brokerem w czasie rzeczywistym.

ID_grupy_JMS

Typ danych:
Łańcuch

Właściwość:
Komunikat

Identyfikator grupy komunikatów, do której należy komunikat.

Ta właściwość jest zdefiniowaną właściwością JMS o nazwie JMS JMSXGroupID. Więcej informacji na temat tej właściwości zawiera specyfikacja *Java Message Service , wersja 1.1*.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie jest poprawna w przypadku połączenia z brokerem w czasie rzeczywistym.

JMSX_GROUPSEQ,

Typ danych:
System.Int32

Właściwość:
Komunikat

Numer kolejny komunikatu w obrębie grupy komunikatów.

Ta właściwość jest właściwością zdefiniowaną przez JMS o nazwie JMS JMSXGroupSeq. Więcej informacji na temat tej właściwości zawiera specyfikacja *Java Message Service , wersja 1.1*.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie jest poprawna w przypadku połączenia z brokerem w czasie rzeczywistym.

ID_UŻYTKOWNIKA_JMS

Typ danych:
Łańcuch

Właściwość:
Komunikat

Identyfikator użytkownika powiązany z aplikacją, która wysłała komunikat.

Ta właściwość jest właściwością zdefiniowaną przez JMS o nazwie JMS name JMSXUserID. Więcej informacji na temat tej właściwości zawiera specyfikacja *Java Message Service , wersja 1.1*.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość nie jest poprawna w przypadku połączenia z brokerem w czasie rzeczywistym.

XMSC_ASYNC_EXCEPTIONS

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: ASYNCEXCEPTION

Skrócona nazwa narzędzia administracyjnego JMS: AEX

Ta właściwość określa, czy aplikacja XMS ma informować interfejs ExceptionListener tylko wtedy, gdy połączenie zostało zerwane, czy też wtedy, gdy dowolny wyjątek wystąpi asynchronicznie w wywołaniu interfejsu API XMS. Ta właściwość ma zastosowanie względem wszystkich połączeń utworzonych w tej fabryce połączeń, które mają zarejestrowany interfejs ExceptionListener.

Poprawne wartości dla tej właściwości to:

XMSC_ASYNC_EXCEPTIONS_ALL
Wszystkie wyjątki wykryte asynchronicznie, poza zasięgiem synchronicznego wywołania interfejsu API, oraz wszystkie wyjątki zerwania połączenia są wysyłane do obiektu ExceptionListener.

XMSC_ASYNC_EXCEPTIONS_CONNECTIONBROKEN
Do obiektu ExceptionListenersą wysyłane tylko wyjątki wskazujące zerwane połączenie. Wszelkie inne wyjątki występujące podczas przetwarzania asynchronicznego nie są zgłaszane do obiektu ExceptionListener, dlatego aplikacja nie jest informowana o tych wyjątkach.

Domyślnie ta właściwość ma wartość XMSC_ASYNC_EXCEPTIONS_ALL.

XMSC_CLIENT_ID

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: CLIENTID

Skrócona nazwa narzędzia administracyjnego JMS: CID

Identyfikator klienta dla połączenia.

Identyfikator klienta jest używany tylko do obsługi trwałych subskrypcji w domenie publikowania/subskrypcji i jest ignorowany w domenie typu punkt z punktem. Więcej informacji na temat ustawiania identyfikatorów klienta zawiera sekcja ConnectionFactories i obiekty połączenia.

Ta właściwość nie ma zastosowania w przypadku połączenia z brokerem w czasie rzeczywistym.

XMSC_CONNECTION_TYPE

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Typ serwera przesyłania komunikatów, z którym łączy się aplikacja.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_CT_RTT	Połączenie w czasie rzeczywistym z brokerem.
XMSC_CT_WMQ	Połączenie z menedżerem kolejek produktu IBM MQ .
XMSC_CT_WPM	Połączenie z WebSphere Application Server service integration bus.

Domyślnie właściwość nie jest ustawiona.

Tryb dostarczania XMSC_DELIVERY_MODE

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Nazwa używana w identyfikatorze URI:

`persistence` (dla miejsca docelowego IBM MQ)
`deliveryMode` (dla miejsca docelowego domyślnego dostawcy przesyłania komunikatów produktu WebSphere)

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: PERSISTENCE

Skrócona nazwa narzędzia administracyjnego JMS: PER

Tryb dostarczania komunikatów wysyłanych do miejsca docelowego.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_DELIVERY_NOT_PERSISTENT (XMSC_DELIVERY_NOT_PERSISTENT	Komunikat wysłany do miejsca docelowego jest nietrwały. Domyślny tryb dostarczania producenta komunikatów lub dowolny tryb dostarczania określony w wywołaniu Send jest ignorowany. Jeśli miejscem docelowym jest kolejka IBM MQ , wartość atrybutu kolejki *DefPersistence* jest również ignorowana.
XMSC_DELIVERY_PERSISTENT	Komunikat wysłany do miejsca docelowego jest trwały. Domyślny tryb dostarczania producenta komunikatów lub dowolny tryb dostarczania określony w wywołaniu Send jest ignorowany. Jeśli miejscem docelowym jest kolejka IBM MQ , wartość atrybutu kolejki *DefPersistence* jest również ignorowana.

Poprawna wartość	Znaczenie
XMSC_DELIVERY_AS_APP	Komunikat wysłany do miejsca docelowego ma tryb dostarczania określony w wywołaniu Send. Jeśli wywołanie Send nie określa trybu dostarczania, zamiast niego używany jest domyślny tryb dostarczania producenta komunikatów. Jeśli miejscem docelowym jest kolejka IBM MQ , wartość atrybutu kolejki *DefPersistence* jest ignorowana.
XMSC_DELIVERY_AS_DEST	Jeśli miejscem docelowym jest kolejka IBM MQ , tryb dostarczania komunikatu umieszczonego w kolejce jest określony przez wartość atrybutu kolejki *DefPersistence*. Domyślny tryb dostarczania producenta komunikatów lub dowolny tryb dostarczania określony w wywołaniu Send jest ignorowany. Jeśli miejscem docelowym nie jest kolejka IBM MQ , znaczenie jest takie samo jak w przypadku kolejki XMSC_DELIVERY_AS_APP.

Wartością domyślną jest XMSC_DELIVERY_AS_APP.

XMSC_IC_PROVIDER_URL (XMSC_PROVIDER_URL)

Typ danych:
Łańcuch

Właściwość:
InitialContext

Umożliwia wskazanie katalogu nazw JNDI. Dzięki niej usługa nazw COS nie musi znajdować się na tym samym serwerze co usługa WWW.

XMSC_IC_SECURITY_AUTHENTICATION

Typ danych:
Łańcuch

Właściwość:
InitialContext

Na podstawie interfejsu kontekstu Java SECURITY_AUTHENTICATION. Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.

XMSC_IC_SECURITY_CREDENTIALS

Typ danych:
Łańcuch

Właściwość:
InitialContext

W oparciu o Java Context interface SECURITY_CREDENTIALS. Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.

XMSC_IC_SECURITY_PRINCIPAL

Typ danych:
Łańcuch

Właściwość:
InitialContext

W oparciu o interfejs kontekstu Java SECURITY_PRINCIPAL. Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.

XMSC_IC_SECURITY_PROTOCOL

Typ danych:
Łańcuch

Właściwość:
InitialContext

Na podstawie parametru Java Context interface SECURITY_PROTOCOL Ta właściwość ma zastosowanie tylko do kontekstu nazewnictwa COS.

Adres XMSC_IC_URL

Typ danych:
Łańcuch

Właściwość:
InitialContext

W przypadku kontekstów LDAP i FileSystem: adres repozytorium zawierającego obiekty administrowane.

W przypadku kontekstów LDAP i FileSystem: adres repozytorium zawierającego obiekty administrowane.

XMSC_IS_SUBSCRIPTION_MULTICAST

Typ danych:
System.Boolean

Właściwość:
MessageConsumer

Wskazuje, czy komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport. Ta właściwość jest tylko do odczytu.

Właściwość ma wartość true, jeśli komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport. W przeciwnym razie wartością jest false.

Ta właściwość ma zastosowanie tylko w przypadku połączenia z brokerem w czasie rzeczywistym.

XMSC_IS_SUBSCRIPTION_RELIABLE_MULTICAST

Typ danych:
System.Boolean

Właściwość:
MessageConsumer

Wskazuje, czy komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport z niezawodną jakością usługi. Ta właściwość jest tylko do odczytu.

Właściwość ma wartość true, jeśli komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport z niezawodną jakością usługi. W przeciwnym razie wartością jest false.

Ta właściwość ma zastosowanie tylko w przypadku połączenia z brokerem w czasie rzeczywistym.

XMSC_JMS_MAJOR_VERSION

Typ danych:
System.Int32

Właściwość:
Dane ConnectionMeta

Numer wersji głównej specyfikacji JMS , na której opiera się produkt XMS . Ta właściwość jest tylko do odczytu.

XMSC_JMS_MINOR_VERSION

Typ danych:
System.Int32

Właściwość:
Dane ConnectionMeta

Numer wersji podrzędnej specyfikacji JMS , na której opiera się produkt XMS . Ta właściwość jest tylko do odczytu.

XMSC_JMS_VERSION (wersja JMS)

Typ danych:
Łańcuch

Właściwość:
Dane ConnectionMeta

Identyfikator wersji specyfikacji JMS , na której opiera się produkt XMS . Ta właściwość jest tylko do odczytu.

XMSC_MAJOR_VERSION

Typ danych:
System.Int32

Właściwość:
Dane ConnectionMeta

Numer wersji klienta XMS . Ta właściwość jest tylko do odczytu.

XMSC_MINOR_VERSION

Typ danych:
System.Int32

Właściwość:
Dane ConnectionMeta

Numer wersji klienta XMS . Ta właściwość jest tylko do odczytu.

HASŁO XMSC_PASSWORD

Typ danych:
Tablica bajtów

Właściwość:
ConnectionFactory

Hasło, które może być używane do uwierzytelniania aplikacji na etapie podejmowania próby nawiązania połączenia z serwerem przesyłania komunikatów. Hasło jest używane z właściwością XMSC_USERID .

Domyślnie właściwość nie jest ustawiona.

Multi W przypadku nawiązywania połączenia z produktem IBM MQ w systemie Wiele platformi ustawienia właściwości XMSC_USERID fabryki połączeń muszą być zgodne z wartością **userid** zalogowanego użytkownika. Jeśli te właściwości nie zostaną ustawione, menedżer kolejek domyślnie użyje wartości **userid** zalogowanego użytkownika. Jeśli wymagane jest dalsze uwierzytelnianie pojedynczych użytkowników na poziomie połączenia, można napisać wyjście uwierzytelniania klienta

skonfigurowane w programie IBM MQ. Więcej informacji na temat tworzenia wyjścia uwierzytelniania klienta zawiera sekcja Planowanie uwierzytelniania dla aplikacji klienckiej.

z/OS Aby uwierzytelnić użytkownika podczas nawiązywania połączenia z produktem IBM MQ for z/OS , należy użyć wyjścia zabezpieczeń.

*PRIORYTET_XMSC_*

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Nazwa używana w identyfikatorze URI:
priorytet

Priorytet komunikatów wysyłanych do miejsca docelowego.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
Liczba całkowita z zakresu od 0, najniższy priorytet, do 9, najwyższy priorytet	Komunikat wysłany do miejsca docelowego ma określony priorytet. Domyślny priorytet producenta komunikatów lub dowolny priorytet określony w wywołaniu Send jest ignorowany. Jeśli miejscem docelowym jest kolejka IBM MQ , wartość atrybutu kolejki **DefPriority** jest również ignorowana.
XMSC_PRIORITY_AS_APP	Komunikat wysłany do miejsca docelowego ma priorytet określony w wywołaniu Send. Jeśli wywołanie Send nie określa priorytetu, używany jest domyślny priorytet producenta komunikatu. Jeśli miejscem docelowym jest kolejka IBM MQ , wartość atrybutu kolejki **DefPriority** jest ignorowana.
XMSC_PRIORITY_AS_DEST	Jeśli miejscem docelowym jest kolejka IBM MQ , komunikat umieszczony w kolejce ma priorytet określony przez wartość atrybutu kolejki **DefPriority**. Domyślny priorytet producenta komunikatów lub dowolny priorytet określony w wywołaniu Send jest ignorowany. Jeśli miejscem docelowym nie jest kolejka IBM MQ , znaczenie jest takie samo jak w przypadku kolejki XMSC_PRIORITY_AS_APP.

Wartością domyślną jest XMSC_PRIORITY_AS_APP.

WebSphere MQ Real-Time Transport i WebSphere MQ Multicast Transport nie podejmują żadnych działań w zależności od priorytetu komunikatu.

NAZWA_DOSTAWCY XMSC_PROVIDER_NAME

Typ danych:
Łańcuch

Właściwość:
Dane ConnectionMeta

Dostawca klienta XMS . Ta właściwość jest tylko do odczytu.

XMSC_RTT_BROKER_PING_INTERVAL

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Odstęp czasu (w milisekundach), po upływie którego klient XMS .NET sprawdza połączenie z serwerem przesyłania komunikatów w czasie rzeczywistym w celu wykrycia dowolnego działania. Jeśli nie zostanie wykryta żadna aktywność, klient inicjuje komendę ping; połączenie jest zamykane, jeśli nie zostanie wykryta żadna odpowiedź na komendę ping.

Wartością domyślną tej właściwości jest 30000.

XMSC_RTT_CONNECTION_PROTOCOL

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Protokół komunikacyjny używany do nawiązywania połączenia z brokerem w czasie rzeczywistym.

Wartością tej właściwości musi być XMSC_RTT_CP_TCP, co oznacza połączenie w czasie rzeczywistym z brokerem za pośrednictwem protokołu TCP/IP. Wartością domyślną jest XMSC_RTT_CP_TCP.

XMSC_RTT_HOST_NAME

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Miejsce działania brokera: nazwa hosta lub adres IP systemu.

Ta właściwość jest używana razem z właściwością XMSC_RTT_PORT do identyfikowania brokera.

Domyślnie właściwość nie jest ustawiona.

XMSC_RTT_LOCAL_ADDRESS

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa hosta lub adres IP interfejsu sieci lokalnej na potrzeby nawiązania połączenia z brokerem w czasie rzeczywistym.

Ta właściwość jest użyteczna tylko wtedy, gdy system, w którym działa aplikacja, ma dwa lub więcej interfejsów sieciowych, a użytkownik musi mieć możliwość określenia interfejsu, który ma być używany dla połączenia w czasie rzeczywistym. Jeśli system ma tylko jeden interfejs sieciowy, można użyć tylko tego interfejsu. Jeśli system ma dwa lub więcej interfejsów sieciowych, a właściwość nie jest ustawiona, interfejs jest wybierany losowo.

Domyślnie właściwość nie jest ustawiona.

XMSC_RTT_MULTICAST (MULTICAST)

Typ danych:
System.Int32

Właściwość:
ConnectionFactory i Destination

Nazwa używana w identyfikatorze URI:
mulicast

Ustawienie rozsyłania grupowego dla fabryki połączeń lub miejsca docelowego. Ta właściwość może mieć tylko miejsce docelowe, które jest tematem.

Aplikacja używa tej właściwości w celu włączenia rozsyłania grupowego w powiązaniu z połączeniem czasu rzeczywistego z brokerem oraz, jeśli rozsyłanie grupowe jest włączone, w celu określenia dokładnego sposobu, w jaki rozsyłanie grupowe jest używane do dostarczania komunikatów z brokera do konsumenta komunikatów. Ta właściwość nie ma wpływu na sposób, w jaki producent komunikatów wysyła komunikaty do brokera.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_RTT_MULTICAST_DISABLED,	Komunikaty nie są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport. Ta wartość jest wartością domyślną dla obiektu ConnectionFactory .
XMSC_RTT_MULTICAST_ASCF,	Komunikaty są dostarczane do konsumenta komunikatów zgodnie z ustawieniem rozsyłania dla fabryki połączeń powiązanej z konsumentem komunikatów. Ustawienie rozsyłania grupowego dla fabryki połączeń jest odnotowywana podczas tworzenia połączenia. Ta wartość jest poprawna tylko dla obiektu docelowego i jest wartością domyślną dla obiektu docelowego.
XMSC_RTT_MULTICAST_ENABLED	Jeśli temat jest skonfigurowany do rozsyłania grupowego w brokerze, komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport. Niezawodna jakość usługi jest używana, jeśli temat jest skonfigurowany do niezawodnego rozsyłania grupowego.
XMSC_RTT_MULTICAST_RELIABLE (niezawodny)	Jeśli temat jest skonfigurowany na potrzeby niezawodnego rozsyłania grupowego w brokerze, komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport z niezawodną jakością usługi. Jeśli temat nie jest skonfigurowany do niezawodnego rozsyłania grupowego, nie można utworzyć konsumenta komunikatów dla tematu.
XMSC_RTT_MULTICAST_NOT_RELIABLE,	Jeśli temat jest skonfigurowany do rozsyłania grupowego w brokerze, komunikaty są dostarczane do konsumenta komunikatów przy użyciu produktu WebSphere MQ Multicast Transport. Niezawodna jakość usługi nie jest używana, nawet jeśli temat jest skonfigurowany do niezawodnego rozsyłania grupowego.

Port RTT_XMSC_PORT

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Numer portu, na którym broker nasłuchuje żądań przychodzących. W brokerze należy skonfigurować węzeł przetwarzania komunikatów przepływu Real-timeInput lub Real-timeOptimized, aby nasłuchiwał na tym porcie.

Ta właściwość jest używana razem z właściwością XMSC_RTT_HOST_NAME do identyfikowania brokera.

Wartością domyślną tej właściwości jest XMSC_RTT_DEFAULT_PORT lub 1506.

XMSC_TIME_TO_LIVE (czas życia)

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Nazwa używana w identyfikatorze URI:

Utrata ważności (dla miejsca docelowego IBM MQ)
timeToLive (dla domyślnego miejsca docelowego dostawcy przesyłania komunikatów produktu WebSphere)

Czas życia komunikatów wysyłanych do miejsca docelowego.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
0	Komunikat wysłany do miejsca docelowego nigdy nie traci ważności.
Dodatnia liczba całkowita	Komunikat wysłany do miejsca docelowego ma określony czas życia (w milisekundach). Domyślny czas życia producenta komunikatu lub czas życia określony w wywołaniu Send jest ignorowany.
XMSC_TIME_TO_LIVE_AS_APP (czas oczekiwania na live_as_app)	Komunikat wysłany do miejsca docelowego ma czas życia określony w wywołaniu Send. Jeśli wywołanie Send nie określa czasu życia, zostanie użyty domyślny czas życia producenta komunikatu.

Wartością domyślną jest XMSC_TIME_TO_LIVE_AS_APP.

ID_UŻYTKOWNIKA_XMSC_ID

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Identyfikator użytkownika, który może być używany do uwierzytelniania aplikacji na etapie podejmowania próby nawiązania połączenia z serwerem przesyłania komunikatów. Identyfikator użytkownika jest używany z właściwością XMSC_PASSWORD .

Domyślnie właściwość nie jest ustawiona.

Multi W przypadku nawiązywania połączenia z produktem IBM MQ for Multiplatformsi ustawienia właściwości XMSC_USERID fabryki połączeń muszą być zgodne z wartością **userid** zalogowanego użytkownika. Jeśli te właściwości nie zostaną ustawione, menedżer kolejek domyślnie użyje wartości **userid** zalogowanego użytkownika. Jeśli wymagane jest dalsze uwierzytelnianie pojedynczych użytkowników na poziomie połączenia, można napisać wyjście uwierzytelniania klienta skonfigurowane w programie IBM MQ. Więcej informacji na temat tworzenia wyjścia uwierzytelniania klienta zawiera sekcja Planowanie uwierzytelniania dla aplikacji klienckiej.

z/OS Aby uwierzytelnić użytkownika podczas nawiązywania połączenia z produktem IBM MQ for z/OS , należy użyć wyjścia zabezpieczeń.

WERSJA_XMSC_VERSION

Typ danych:
Łańcuch

Właściwość:
Dane ConnectionMeta

Identyfikator wersjiXMSklienta. Ta właściwość jest tylko do odczytu.

V 9.3.0 *XMSC_WMQ_BALANCING_APPLICATION_TYPE*

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_BALANCING_APPLICATION_TYPE_SIMPLE	Proste równoważenie; oprócz reguł opisanych w sekcji Wpływanie na ponowne równoważenie aplikacji w jednolitych klastrachnie są stosowane żadne szczegółowe reguły. Jest to wartość domyślna.
XMSC_WMQ_BALANCING_APPLICATION_TYPE_REQUEST_REPLY	Równoważenie żądanie-odpowiedź. Po każdym wywołaniu MQPUT oczekiwane jest zgodne wywołanie MQGET dla komunikatu odpowiedzi. Równoważenie jest opóźniane do momentu odebrania takiego komunikatu lub przekroczenia limitu czasu komunikatu żądania.

Dodatkowo tę właściwość można ustawić w pliku `client.ini` . Preferowany porządek jest następujący:

1. Właściwości ustawione w aplikacji
2. Zgodna sekcja aplikacji w pliku `mqclient.ini` .
3. Sekcja wartości domyślnych aplikacji w pliku `mqclient.ini` .

V 9.3.0 *XMSC_WMQ_BALANCING_OPCJE*

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Poprawne wartości właściwości są następujące:

Poprawna wartość	Odpowiednia wartość
XMSC_WMQ_BALANCING_OPTIONS_NONE	Nie ustawiono żadnych opcji. Jest to wartość domyślna
XMSC_WMQ_BALANCING_OPTIONS_IGNORE_TRANSACTION	Ustawienie tej opcji umożliwia ponowne równoważenie aplikacji, nawet jeśli są one w trakcie transakcji.

Dodatkowo tę właściwość można ustawić w pliku `client.ini` . Preferowany porządek jest następujący:

1. Właściwości ustawione w aplikacji
2. Zgodna sekcja aplikacji w pliku `mqclient.ini` .
3. Sekcja wartości domyślnych aplikacji w pliku `mqclient.ini` .

V 9.3.0 *XMSC_WMQ_BALANCING_TIMEOUT (XMSC_W_BALANCING_TIMEOUT)*

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_BALANCING_TIMEOUT_IMMEDIATE	Natychmiastowe przekroczenie limitu czasu
XMSC_WMQ_BALANCING_TIMEOUT_AS_DEFAULT	Ustawiona domyślna wartość limitu czasu. Jest to wartość domyślna
XMSC_WMQ_BALANCING_TIMEOUT_NEVER	Brak limitu czasu

Uwaga: Należy podać tylko jedną wartość ze zdefiniowanych wartości lub wartość z zakresu od 0 do 999999999 sekund.

Dodatkowo tę właściwość można ustawić w pliku `client.ini` . Preferowany porządek jest następujący:

1. Właściwości ustawione w aplikacji
2. Zgodna sekcja aplikacji w pliku `mqclient.ini` .
3. Sekcja wartości domyślnych aplikacji w pliku `mqclient.ini` .

XMSC_WMQ_BROKER_CONTROLQ

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa kolejki sterującej używanej przez broker.

Wartością domyślną tej właściwości jest SYSTEM.BROKER.CONTROL.QUEUE.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WMQ_BROKER_PUBQ

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa kolejki monitorowanej przez broker, do której aplikacje wysyłają publikowane komunikaty.

Wartością domyślną tej właściwości jest SYSTEM.BROKER.DEFAULT.STREAM.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WMQ_BROKER_QMGR

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa menedżera kolejek, z którym broker nawiązał połączenie.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WMQ_BROKER_SUBQ

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa kolejki subskrybenta dla konsumenta nietrwałych komunikatów.

Nazwa kolejki subskrybenta musi zaczynać się od następujących znaków:

SYSTEM.JMS.ND.

Jeśli wszyscy konsumenci nietrwałych komunikatów mają współużytkować kolejkę subskrybenta, należy podać pełną nazwę kolejki współużytkowanej. Zanim aplikacja będzie mogła utworzyć konsument nietrwałych komunikatów, musi istnieć kolejka o podanej nazwie.

Jeśli każdy konsument nietrwałych komunikatów ma pobierać komunikaty z własnej wyłącznej kolejki subskrybenta, należy podać nazwę kolejki, która kończy się gwiazdką (*). Następnie, gdy aplikacja tworzy nietrwałego konsumenta komunikatów, klient XMS tworzy kolejkę dynamiczną do wyłącznego użytku przez konsument komunikatów. Klient XMS używa wartości właściwości do ustawienia zawartości pola **DynamicQName** w deskryptorze obiektu używanym do tworzenia kolejki dynamicznej.

Wartością domyślną tej właściwości jest SYSTEM.JMS.ND.SUBSCRIBER.QUEUE, co oznacza, że produkt XMS domyślnie korzysta z kolejki współużytkowanej.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WMQ_BROKER_VERSION

Typ danych:
System.Int32

Właściwość:
ConnectionFactory i Destination

Nazwa używana w identyfikatorze URI:
brokerVersion

Typ brokera używany na potrzeby połączenia lub miejsca docelowego. Ta właściwość może mieć tylko miejsce docelowe, które jest tematem.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_BROKER_V1	Aplikacja używa brokera publikowania/subskrypcji produktu IBM MQ . Aplikacja może również użyć tej wartości, jeśli wykonywana jest migracja z produktu IBM MQ publish/subscribe do produktu WebSphere Message Broker , ale aplikacja nie została zmieniona.
XMSC_WMQ_BROKER_V2	Aplikacja używa brokera IBM Integration Bus.

Poprawna wartość	Znaczenie
XMSC_WMQ_BROKER_UNSPECIFIED	Po przeprowadzeniu migracji brokera należy ustawić tę właściwość, aby nagłówki RFH2 nie były już używane. Po migracji ta właściwość nie jest już istotna.

Wartość domyślna fabryki połączeń to XMSC_WMQ_BROKER_UNSPECIFIED, ale domyślnie właściwość nie jest ustawiona dla miejsca docelowego. Ustawienie właściwości dla miejsca docelowego powoduje przesłonięcie dowolnej wartości określonej przez właściwość fabryki połączeń.

XMSC_WMQ_CCDTURL

Typ danych:
System.String

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: CCDTURL

Skrócona nazwa narzędzia administracyjnego JMS: CCDT

Adres URL identyfikujący nazwę i położenie pliku zawierającego tabelę definicji kanałów klienta oraz określający sposób dostępu do pliku.

Domyślnie ta właściwość nie jest ustawiona.

XMSC_WMQ_CCSID

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Nazwa używana w identyfikatorze URI:
CCSID

Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, w którym znajdują się łańcuchy danych znakowych w treści komunikatu, gdy klient XMS przekazuje komunikat do miejsca docelowego. Jeśli ta właściwość jest ustawiona dla pojedynczego komunikatu, właściwość JMS_IBM_CHARACTER_SET zastępuje identyfikator CCSID określony dla miejsca docelowego przez tę właściwość.

Wartością domyślną tej właściwości jest 1208.

Ta właściwość dotyczy tylko komunikatów wysyłanych do miejsca docelowego, a nie komunikatów odbieranych z miejsca docelowego.

XMSC_WMQ_CHANNEL

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: CHANNEL

Skrócona nazwa narzędzia administracyjnego JMS: CHAN

Nazwa kanału używanego do nawiązywania połączenia.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta.

XMSC_WMQ_CLIENT_RECONNECT_OPTIONS

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: CLIENTRECONNECTOPTIONS

Skrócona nazwa narzędzia administracyjnego JMS: CROPT

Ta właściwość określa opcje ponownego połączenia klienta dla nowych połączeń utworzonych przez tę fabrykę. Znajduje się w XMSC i ma jedną z następujących wartości:

- WMQ_CLIENT_RECONNECT_AS_DEF (wartość domyślna). Użyj wartości określonej w pliku `mqclient.ini` . Ustaw wartość za pomocą właściwości **DefRecon** w sekcji Channels. Można ją ustawić na jedną z następujących wartości:
  1. Tak. Zachowuje się jak opcja WMQ_CLIENT_RECONNECT
  2. Nie. Opcja domyślna. Nie określa żadnych opcji ponownego połączenia
  3. QMGR. Działa jak opcja WMQ_CLIENT_RECONNECT_Q_MGR
  4. Wyłączone. Działa jak opcja WMQ_CLIENT_RECONNECT_DISABLED
- WMQ_CLIENT_RECONNECT. Ponownie nawiąż połączenie z dowolnym menedżerem kolejek określonym na liście nazw połączeń.
- WMQ_CLIENT_RECONNECT_Q_MGR. Ponownie nawiązuje połączenie z tym samym menedżerem kolejek, z którym jest pierwotnie połączony. Zwraca wartość `MQRC_RECONNECT_QMID_MISMATCH` , jeśli menedżer kolejek, z którym próbuje nawiązać połączenie (określony na liście nazw połączeń), ma inny identyfikator QMID niż menedżer kolejek, z którym pierwotnie nawiązał połączenie.
- WMQ_CLIENT_RECONNECT_DISABLED (WMQ_CLIENT_RECONNECT_DISABLED) Ponowne połączenie jest wyłączone.

XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT,

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: CLIENTRECONNECTTIMEOUT

Skrócona nazwa narzędzia administracyjnego JMS: CRT

Właściwość XMSC_WMQ_CLIENT_RECONNECT_TIMEOUT jest poprawna tylko dla zarządzanego klienta XMS .NET .

Ta właściwość określa czas (w sekundach), przez jaki połączenie klienta próbuje ponownie nawiązać połączenie.

Po próbie ponownego nawiązania połączenia przez ten czas klient zakończy działanie niepowodzeniem z błędem MQRC_RECONNECT_FAILED. Domyślnym ustawieniem tej właściwości jest XMSC.WMQ_CLIENT_RECONNECT_TIMEOUT_DEFAULT.

Wartością domyślną tej właściwości jest 1800.

XMSC_WMQ_CONNECTION_MODE

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Tryb, przy użyciu którego aplikacja nawiązuje połączenie z menedżerem kolejek.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_CM_BINDINGS	Połączenie z menedżerem kolejek w trybie powiązań w celu uzyskania optymalnej wydajności. Ta wartość jest wartością domyślną dla C/C + +.
XMSC_WMQ_CM_CLIENT	Połączenie z menedżerem kolejek w trybie klienta w celu zapewnienia w pełni zarządzanego stosu. Jest to wartość domyślna dla środowiska .NET.
XMSC_WMQ_CM_CLIENT_UNMANAGED (tylko dla środowiska .NET)	Połączenie z menedżerem kolejek, które wymusza niezarządzany stos klienta.

XMSC_WMQ_CONNECTION_NAME_LIST

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: CONNECTIONNAMELIST

Skrócona nazwa narzędzia administracyjnego JMS: CNLIST

Ta właściwość określa hosty, z którymi klient próbuje ponownie nawiązać połączenie po zerwaniu połączenia.

Lista nazw połączeń jest listą rozdzielonych przecinkami par host/port IP. Domyślnym ustawieniem tej właściwości jest WMQ_CONNECTION_NAME_LIST_DEFAULT.

Na przykład`127.0.0.1 (1414),host2.example.com(1400)`

Domyślnym ustawieniem tej właściwości jest `localhost (1414)`.

XMSC_WMQ_DUR_SUBQ

Typ danych:
Łańcuch

Właściwość:
Miejsce docelowe

Nazwa kolejki trwałego subskrybenta, który odbiera komunikaty z miejsca docelowego. Ta właściwość może mieć tylko miejsce docelowe, które jest tematem.

Nazwa kolejki subskrybenta musi zaczynać się od następujących znaków:

SYSTEM.JMS.D.

Jeśli wszyscy subskrybenci mają współużytkować kolejkę subskrybentów, należy podać pełną nazwę kolejki współużytkowanej. Zanim aplikacja będzie mogła utworzyć trwały subskrybent, musi istnieć kolejka o podanej nazwie.

Jeśli każdy trwały subskrybent ma pobierać komunikaty z własnej wyłącznej kolejki subskrybenta, należy podać nazwę kolejki, która kończy się gwiazdką (*). Następnie, gdy aplikacja tworzy trwałego

subskrybenta, klient XMS tworzy kolejkę dynamiczną do wyłącznego użytku przez trwały subskrybent. Klient XMS używa wartości właściwości do ustawienia zawartości pola **DynamicQName** w deskryptorze obiektu używanym do tworzenia kolejki dynamicznej.

Wartością domyślną tej właściwości jest SYSTEM.JMS.D.SUBSCRIBER.QUEUE, co oznacza, że produkt XMS domyślnie korzysta z kolejki współużytkowanej.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WMQ_ENCODING

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Sposób reprezentowania danych liczbowych w treści komunikatu, gdy klient XMS przekazuje komunikat do miejsca docelowego. W przypadku ustawienia dla pojedynczego komunikatu właściwość JMS_IBM_ENCODING zastępuje kodowanie określone dla miejsca docelowego przez tę właściwość. Ta właściwość określa reprezentację binarnych liczb całkowitych, upakowanych liczb dziesiętnych i liczb zmiennopozycyjnych.

Poprawne wartości właściwości są takie same, jak wartości, które można określić w polu **Encoding** deskryptora komunikatu.

Aplikacja może użyć następujących stałych nazwanych w celu ustawienia właściwości:

Stała nazwana	Znaczenie
MQENC_INTEGER_NORMAL	Normalne kodowanie liczb całkowitych
MQENC_INTEGER_REVERSED,	Kodowanie odwrotnej liczby całkowitej
MQENC_DECIMAL_NORMAL	Normalne kodowanie upakowanych liczb dziesiętnych
MQENC_DECIMAL_REVERSED (odwrócone)	Odwrócone upakowane kodowanie dziesiętne
MQENC_FLOAT_IEEE_NORMAL	Normalne kodowanie zmiennopozycyjne IEEE
MQENC_FLOAT_IEEE_REVERSED	Odwrotne kodowanie zmiennopozycyjne IEEE
MQENC_FLOAT_S390	Kodowanie zmiennopozycyjne architektury z/OS
RODZIMA MQENC	Kodowanie na komputerze rodzimym

Aby utworzyć wartość właściwości, aplikacja może dodać trzy z następujących stałych:

- Stała, której nazwa rozpoczyna się od MQENC_INTEGER, służąca do określania reprezentacji binarnych liczb całkowitych.
- Stała, której nazwa rozpoczyna się od MQENC_DECIMAL, służąca do określania reprezentacji upakowanych dziesiętnych liczb całkowitych.
- Stała, której nazwa rozpoczyna się od MQENC_FLOAT, służąca do określania reprezentacji liczb zmiennopozycyjnych.

Alternatywnie aplikacja może ustawić tę właściwość na wartość MQENC_NATIVE, której wartość jest zależna od środowiska.

Wartością domyślną tej właściwości jest MQENC_NATIVE.

Ta właściwość dotyczy tylko komunikatów wysyłanych do miejsca docelowego, a nie komunikatów odbieranych z miejsca docelowego.

XMSC_WMQ_FAIL_IF_QUIESCE

Typ danych:
System.Int32

Właściwość:
ConnectionFactory i Destination

Nazwa używana w identyfikatorze URI:
FAILIFQUIESCE

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: FAILIFQUIESCE

Skrócona nazwa narzędzia administracyjnego JMS: FIQ

Określa, czy wywołania pewnych metod nie powiodą się, jeśli menedżer kolejek, z którym aplikacja nawiązała połączenie, będzie w stanie wyciszania.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_FIQ_YES	Wywołania niektórych metod nie powiodą się, jeśli menedżer kolejek jest w stanie wyciszania. Gdy aplikacja wykryje, że menedżer kolejek jest wyciszany, może zakończyć swoje natychmiastowe zadanie i zamknąć połączenie, umożliwiając zatrzymanie menedżera kolejek.
XMSC_WMQ_FIQ_NO	Żadne wywołania metody nie powiodły się, ponieważ menedżer kolejek jest w stanie wyciszania. Jeśli ta wartość zostanie określona, aplikacja nie może wykryć, że menedżer kolejek jest wyciszany. Aplikacja może kontynuować wykonywanie operacji na menedżerze kolejek, co uniemożliwi zatrzymanie menedżera kolejek.

Wartością domyślną dla fabryki połączeń jest XMSC_WMQ_FIQ_YES, ale domyślnie właściwość nie jest ustawiona dla miejsca docelowego. Ustawienie właściwości dla miejsca docelowego powoduje przesłonięcie dowolnej wartości określonej przez właściwość fabryki połączeń.

XMSC_WMQ_MESSAGE_BODY

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Ta właściwość określa, czy aplikacja XMS przetwarza MQRFH2 komunikatu IBM MQ jako część ładunku komunikatu (czyli jako część treści komunikatu).

Uwaga: Podczas wysyłania komunikatów do miejsca docelowego właściwość XMSC_WMQ_MESSAGE_BODY zastępuje istniejącą właściwość miejsca docelowego XMS XMSC_WMQ_TARGET_CLIENT.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_MESSAGE_BODY_JMS
Odbierz: typ i treść komunikatu przychodzącego XMS są określone przez treść nagłówka MQRFH2 (jeśli istnieje) lub deskryptora MQMD (jeśli nie ma nagłówka MQRFH2) w odebranym komunikacie IBM MQ .

Wyślij: Treść komunikatu wychodzącego XMS zawiera wstępnie i automatycznie wygenerowany nagłówek MQRFH2 na podstawie właściwości komunikatu XMS i pól nagłówka.

XMSC_WMQ_MESSAGE_BODY_MQ
Odbierz: typ komunikatu przychodzącego XMS to zawsze ByteMessage, niezależnie od treści odebranego komunikatu IBM MQ lub pola formatu odebranego deskryptora MQMD. Treść komunikatu XMS to niezmienione dane komunikatu zwracane przez bazowe wywołanie interfejsu API dostawcy przesyłania komunikatów. Zestaw znaków i kodowanie danych w treści komunikatu są określane przez pola CodedCharSetId i Encoding w strukturze MQMD. Format danych w treści komunikatu jest określany przez pole Format deskryptora MQMD.

Wyślij: treść komunikatu wychodzącego XMS zawiera ładunek aplikacji w postaci, w jakiej jest; do treści nie jest dodawany automatycznie generowany nagłówek IBM MQ .

XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED
Odbierz: klient XMS określa odpowiednią wartość dla tej właściwości. W ścieżce odbiorczej ta wartość jest wartością właściwości WMQ_MESSAGE_BODY_JMS.

Wyślij: klient XMS określa odpowiednią wartość dla tej właściwości. W przypadku ścieżki wysyłania ta wartość jest wartością właściwości XMSC_WMQ_TARGET_CLIENT.

Domyślnie ta właściwość ma wartość XMSC_WMQ_MESSAGE_BODY_UNSPECIFIED.

XMSC_WMQ_MQMD_MESSAGE_CONTEXT

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Określa poziom kontekstu komunikatu, który ma zostać ustawiony przez aplikację XMS . Aby ta właściwość miała zastosowanie, aplikacja musi być uruchamiana z odpowiednimi uprawnieniami dotyczącymi kontekstu.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_MDCTX_DEFAULT
W przypadku komunikatów wychodzących wywołanie API MQOPEN i struktura MQPMO nie określają jawnych opcji kontekstu komunikatu.

XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT
Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_IDENTITY_CONTEXT, a struktura MQPMO określa opcję MQPMO_SET_IDENTITY_CONTEXT.

XMSC_WMQ_MDCTX_SET_ALL_CONTEXT
Wywołanie funkcji API MQOPEN określa opcję kontekstu komunikatu MQOO_SET_ALL_CONTEXT, a struktura MQPMO określa opcję MQPMO_SET_ALL_CONTEXT.

Domyślnie ta właściwość ma wartość XMSC_WMQ_MDCTX_DEFAULT.

Uwaga: Ta właściwość nie ma zastosowania, gdy aplikacja nawiązuje połączenie z produktem WebSphere Application Server service integration bus.

Następujące właściwości wymagają, aby właściwość XMSC_WMQ_MQMD_MESSAGE_CONTEXT była ustawiona na wartość właściwości XMSC_WMQ_MDCTX_SET_IDENTITY_CONTEXT lub XMSC_WMQ_MDCTX_SET_ALL_CONTEXT podczas wysyłania komunikatu w celu uzyskania pożądanego efektu:

- JMS_IBM_MQMD_USERIDENTIFIER
- JMS_IBM_MQMD_ACCOUNTINGTOKEN,
- JMS_IBM_MQMD_APPLIDENTITYDATA

Następujące właściwości wymagają ustawienia właściwości XMSC_WMQ_MQMD_MESSAGE_CONTEXT na wartość właściwości XMSC_WMQ_MDCTX_SET_ALL_CONTEXT podczas wysyłania komunikatu w celu uzyskania efektu:

- JMS_IBM_MQMD_PUTAPPLTYPE
- JMS_IBM_MQMD_PUTAPPLNAME
- JMS_IBM_MQMD_PUTDATE
- JMS_IBM_MQMD_PUTTIME
- JMS_IBM_MQMD_APPLORIGINDATA

XMSC_WMQ_MQMD_READ_ENABLED

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Ta właściwość określa, czy aplikacja XMS może wyodrębniać wartości pól MQMD, czy nie.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_READ_ENABLED_NO
Podczas wysyłania komunikatów właściwości JMS_IBM_MQMD* wysłanego komunikatu nie są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD.

Podczas odbierania komunikatów żadna z właściwości JMS_IBM_MQMD* nie jest dostępna w odebranym komunikacie, nawet jeśli niektóre lub wszystkie z nich są ustawione przez nadawcę.

XMSC_WMQ_READ_ENABLED_YES
Podczas wysyłania komunikatów wszystkie właściwości JMS_IBM_MQMD* w wysłanym komunikacie są aktualizowane w celu odzwierciedlenia zaktualizowanych wartości pól w strukturze MQMD, w tym również te właściwości, które nie zostały jawnie ustawione przez nadawcę.

Podczas odbierania komunikatów wszystkie właściwości JMS_IBM_MQMD* są dostępne w odebranym komunikacie, w tym te właściwości, które nie zostały jawnie ustawione przez nadawcę.

Domyślnie ta właściwość ma wartość XMSC_WMQ_READ_ENABLED_NO.

XMSC_WMQ_MQMD_WRITE_ENABLED

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Ta właściwość określa, czy aplikacja XMS może ustawiać wartości pól MQMD.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_WRITE_ENABLED_NO
Wszystkie właściwości JMS_IBM_MQMD* są ignorowane, a ich wartości nie są kopiowane do bazowej struktury MQMD.

XMSC_WMQ_WRITE_ENABLED_YES
Właściwości JMS_IBM_MQMD* są przetwarzane. Ich wartości są kopiowane do bazowej struktury MQMD.

Domyślnie ta właściwość ma wartość XMSC_WMQ_WRITE_ENABLED_NO.

XMSC_WMQ_PUT_ASYNC_ALLOWED

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Ta właściwość określa, czy producenci komunikatów mogą używać asynchronicznych operacji put w celu wysyłania komunikatów do tego miejsca docelowego.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST
Określ, czy asynchroniczne operacje umieszczania są dozwolone, odwołując się do definicji kolejki lub tematu.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_Q_DEF
Określ, czy asynchroniczne operacje umieszczania są dozwolone, odwołując się do definicji kolejki.

XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_TOPIC_DEF
Określ, czy dozwolone są asynchroniczne operacje umieszczania, odwołując się do definicji tematu.

XMSC_WMQ _PUT_ASYNC_ALLOWED_DISABLED
Asynchroniczne operacje umieszczania nie są dozwolone.

XMSC_WMQ _PUT_ASYNC_ALLOWED_ENABLED
Asynchroniczne operacje umieszczania są dozwolone.

Domyślnie ta właściwość ma wartość XMSC_WMQ _PUT_ASYNC_ALLOWED_AS_DEST.

Uwaga: Ta właściwość nie ma zastosowania, gdy aplikacja łączy się z produktem WebSphere Application Server service integration bus.

XMSC_WMQ_READ_AHEAD_ALLOWED

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Ta właściwość określa, czy konsumenty komunikatów i przeglądarki kolejek mogą używać operacji odczytu z wyprzedzeniem do pobierania nietrwałych komunikatów nietransakcyjnych z tego miejsca docelowego do buforu wewnętrznego przed ich odebraniem.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_Q_DEF
Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_ TEMAT _DEF
Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji tematu.

XMSC_WMQ_READ_AHEAD_ALLOWED_AS_DEST
Określ, czy odczyt z wyprzedzeniem jest dozwolony, odwołując się do definicji kolejki lub tematu.

XMSC_WMQ_READ_AHEAD_ALLOWED_DISABLED
Odczyt z wyprzedzeniem jest niedozwolony podczas odbierania lub przeglądania komunikatów.

XMSC_WMQ_READ_AHEAD_ALLOWED_ENABLED
Odczyt z wyprzedzeniem jest dozwolony.

Domyślnie ta właściwość ma wartość XMSC_WMQ _READ_AHEAD_ALLOWED_AS_DEST.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

W przypadku komunikatów dostarczanych do asynchronicznego procesu nasłuchującego komunikatów ta właściwość określa, co stanie się z komunikatami w wewnętrznym buforze odczytu z wyprzedzeniem, gdy konsument komunikatów zostanie zamknięty.

Ta właściwość ma zastosowanie w przypadku określania opcji zamykania kolejki podczas odbierania komunikatów z miejsca docelowego i nie ma zastosowania w przypadku wysyłania komunikatów do miejsca docelowego.

Ta właściwość jest ignorowana w przeglądarkach kolejek, ponieważ podczas przeglądania komunikaty są nadal dostępne w kolejkach.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT
Tylko bieżące wywołanie obiektu nasłuchiwania komunikatów kończy się przed zwróceniem, potencjalnie pozostawiając komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem, które są następnie usuwane.

XMSC_WMQ_READ_AHEAD_CLOSE_POLICY_DELIVER_ALL
Wszystkie komunikaty w wewnętrznym buforze odczytu z wyprzedzeniem są dostarczane do obiektu nasłuchiwania komunikatów aplikacji przed powrotem.

Domyślnie ta właściwość ma wartość XMSC_WMQ _READ_AHEAD_CLOSE_POLICY_DELIVER_CURRENT.

Uwaga:

Nieprawidłowe zakończenie aplikacji
Wszystkie komunikaty w buforze odczytu z wyprzedzeniem są tracone po nagłym zakończeniu działania aplikacji XMS .

Konsekwencje dla transakcji
Odczyt z wyprzedzeniem jest wyłączany, gdy aplikacje używają transakcji. Oznacza to, że aplikacja nie widzi żadnych różnic w zachowaniu, gdy używa sesji transakcyjnych.

Konsekwencje trybów oceny sesji
Odczyt z wyprzedzeniem jest włączany w sesji bez transakcji, gdy trybem potwierdzenia jest XMSC_AUTO_ACKNOWLEDGE lub XMSC_DUPS_OK_ACKNOWLEDGE. Odczyt z wyprzedzeniem jest wyłączony, jeśli tryb potwierdzenia sesji to XMSC_CLIENT_ACKNOWLEDGE bez względu na sesje transakcyjne lub nietransakcyjne.

Implikacje dla przeglądarek kolejek i selektorów przeglądarek kolejek
Przeglądarki kolejek i selektory przeglądarek kolejek, używane w aplikacjach XMS , uzyskują przewagę wydajności dzięki odczycie z wyprzedzeniem. Zamknięcie przeglądarki kolejek nie obniża wydajności, ponieważ komunikat jest nadal dostępny w kolejce na potrzeby kolejnych operacji. Nie ma innych konsekwencji dla przeglądarek kolejek i selektorów przeglądarek kolejek, poza korzyściami z wydajności wynikającymi z odczytu z wyprzedzeniem.

XMSC_WMQ_HOST_NAME

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: HOSTNAME

Skrócona nazwa narzędzia administracyjnego JMS: HOST

Miejsce działania menedżera kolejek: nazwa hosta lub adres IP systemu.

Ta właściwość jest używana tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta. Właściwość ta jest używana wraz z właściwością XMSC_WMQ_PORT do identyfikowania menedżera kolejek.

Wartością domyślną tej właściwości jest `localhost`.

XMSC_WMQ_LOCAL_ADDRESS

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: LOCALADDRESS

Skrócona nazwa narzędzia administracyjnego JMS: LA

W przypadku połączenia z menedżerem kolejek: ta właściwość określa używaną wartość interfejsu sieci lokalnej albo portu lokalnego (lub zakresu portów lokalnych) bądź obie te wartości.

Wartością właściwości jest łańcuch w następującym formacie:

[*nazwa_hosta*] [(*low_port*) [,*port_wysoki*])]

Znaczenie zmiennych jest następujące:

nazwa_hosta
Nazwa hosta lub adres IP lokalnego interfejsu sieciowego, który ma być używany dla połączenia.

Podanie tych informacji jest konieczne tylko wtedy, gdy system, w którym działa aplikacja, ma dwa lub więcej interfejsów sieciowych, a użytkownik musi mieć możliwość określenia interfejsu, który ma być używany dla połączenia. Jeśli system ma tylko jeden interfejs sieciowy, można użyć tylko tego interfejsu. Jeśli system ma dwa lub więcej interfejsów sieciowych i nie określono interfejsu, który ma być używany, interfejs jest wybierany losowo.

low_port (low_port)
Numer portu lokalnego, który ma być używany dla połączenia.

Jeśli określono również wartość *high_port* , wartość *low_port* jest interpretowana jako najniższy numer portu z zakresu numerów portów.

port_wysoki
Najwyższy numer portu w zakresie numerów portów. Dla połączenia należy użyć jednego z portów z podanego zakresu.

Maksymalna długość łańcucha wynosi 48 znaków.

Poniżej przedstawiono przykłady poprawnych wartości właściwości:

JUPITER
9.20.4.98
JUPITER (1000)
9.20.4.98(1000,2000)
(1000)
(1000,2000)

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta.

XMSC_WMQ_MESSAGE_SELECTION

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Określa, czy wybór komunikatów jest dokonywany przez klienta XMS , czy przez broker.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_MSEL_CLIENT	Wybór komunikatów jest dokonywany przez klienta XMS .
XMSC_WMQ_MSEL_BROKER	Wybór komunikatów jest dokonywany przez broker.

Wartością domyślną jest XMSC_WMQ_MSEL_CLIENT.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji. Wybór komunikatów przez broker nie jest obsługiwany, jeśli właściwość XMSC_WMQ_BROKER_VERSION jest ustawiona na wartość XMSC_WMQ_BROKER_V1.

XMSC_WMQ_MSG_BATCH_SIZE

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Maksymalna liczba komunikatów pobieranych z kolejki w jednej partii, gdy komunikaty są dostarczane w trybie asynchronicznym.

Jeśli aplikacja używa asynchronicznego dostarczania komunikatów, w pewnych warunkach klient XMS pobiera partię komunikatów z kolejki przed indywidualnym przekazaniem każdego komunikatu do aplikacji. Ta właściwość określa maksymalną liczbę komunikatów, które mogą znajdować się w zadaniu wsadowym.

Wartością właściwości jest dodatnia liczba całkowita, a wartością domyślną jest 10. Należy rozważyć ustawienie innej wartości właściwości tylko wtedy, gdy występuje konkretny problem z wydajnością, który należy rozwiązać.

Jeśli aplikacja jest połączona z menedżerem kolejek za pośrednictwem sieci, zwiększenie wartości tej właściwości może zmniejszyć narzuty sieci i czasy odpowiedzi, ale zwiększyć ilość pamięci wymaganej do przechowywania komunikatów w systemie klienckim. I odwrotnie, obniżenie wartości tej właściwości może zwiększyć narzuty sieci i czasy odpowiedzi, ale zmniejszyć ilość pamięci wymaganej do przechowywania komunikatów.

XMSC_WMQ_POLLING_INTERVAL

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Jeśli kolejka żadnego procesu nasłuchującego w ramach sesji nie zawiera odpowiedniego komunikatu, jest to maksymalny odstęp czasu (w milisekundach) między kolejnymi próbami pobrania komunikatu z kolejki przez każdy z procesów nasłuchujących komunikatów.

Jeśli często zdarza się, że dla żadnego z obiektów nasłuchiwania komunikatów w sesji nie jest dostępny odpowiedni komunikat, należy rozważyć zwiększenie wartości tej właściwości.

Wartością właściwości jest dodatnia liczba całkowita. Wartość domyślna to 5000.

XMSC_WMQ_PORT

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: PORT

Skrócona nazwa narzędzia administracyjnego JMS: PORT

Numer portu, na którym menedżer kolejek nasłuchuje żądań przychodzących.

Ta właściwość jest używana tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta. Ta właściwość jest używana z właściwością XMSC_WMQ_HOST_NAME do identyfikowania menedżera kolejek.

Wartością domyślną tej właściwości jest XMSC_WMQ_DEFAULT_CLIENT_PORT lub 1414.

XMSC_WMQ_PROVIDER_VERSION

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Wersja, wydanie, poziom modyfikacji i pakiet poprawek menedżera kolejek, z którym aplikacja ma nawiązać połączenie. Poprawne wartości dla tej właściwości to:

- Nieokreślona

Lub łańcuch w jednym z następujących formatów

- V.R.M.F
- V.R.M
- V.R
- V

Gdzie: V, R, M i F są wartościami całkowitymi większymi niż zero lub równymi zero.

Domyślnie ta właściwość jest ustawiona na wartość "unspecified".

Wersja klienta IBM MQ odgrywa również główną rolę w tym, czy aplikacja kliencka XMS może korzystać z konkretnych funkcji IBM MQ .

Uwaga: Właściwość systemowa XMSC_WMQ_OVERRIDEPROVIDERVERSION przesłania właściwość XMSC_WMQ_PROVIDER_VERSION. Tej właściwości można użyć, jeśli nie można zmienić ustawienia fabryki połączeń.

XMSC_WMQ_PUB_ACK_INTERVAL

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Liczba komunikatów opublikowanych przez publikator, zanim klient XMS zażąda potwierdzenia od brokera.

Jeśli wartość tej właściwości zostanie zmniejszona, klient będzie częściej żądać potwierdzeń, co spowoduje zmniejszenie wydajności publikatora. Jeśli wartość zostanie zwiększona, zgłaszanie wyjątku przez klient w przypadku awarii brokera będzie trwać dłużej.

Wartością właściwości jest dodatnia liczba całkowita. Wartość domyślna wynosi 25.

XMSC_WMQ_QMGR_CCSID

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Identyfikator (CCSID) kodowanego zestawu znaków lub strony kodowej, w którym pola danych znakowych zdefiniowane w interfejsie MQI (Message Queue Interface) są wymieniane między klientem XMS a klientem IBM MQ . Ta właściwość nie ma zastosowania do łańcuchów danych znakowych w treści komunikatów.

Gdy aplikacja XMS nawiązuje połączenie z menedżerem kolejek w trybie klienta, klient XMS łączy się z klientem IBM MQ . Informacje wymieniane między dwoma klientami zawierają pola danych znakowych, które są zdefiniowane w MQI. W normalnych okolicznościach klient IBM MQ zakłada, że te pola znajdują się w stronie kodowej systemu, w którym działają klienty. Jeśli klient XMS udostępnia i oczekuje na

odebranie tych pól w innej stronie kodowej, należy ustawić tę właściwość, aby poinformować klienta IBM MQ .

Gdy klient IBM MQ przekazuje te pola danych znakowych do menedżera kolejek, dane w nich zawarte muszą zostać w razie potrzeby przekształcone w stronę kodową używaną przez menedżer kolejek. Podobnie, gdy klient IBM MQ odbiera te pola z menedżera kolejek, w razie potrzeby dane z tych pól muszą zostać przekształcone w stronę kodową, w której klient XMS oczekuje, że otrzyma dane. Klient IBM MQ używa tej właściwości do przeprowadzania konwersji danych.

Domyślnie właściwość nie jest ustawiona.

Ustawienie tej właściwości jest równoważne ustawieniu zmiennej środowiskowej MQCCSID dla klienta IBM MQ , który obsługuje rodzime aplikacje klienckie IBM MQ . Więcej informacji na temat tej zmiennej środowiskowej zawiera sekcja MQCCSID.

XMSC_WMQ_QUEUE_MANAGER

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: QMANAGER

Skrócona nazwa narzędzia administracyjnego JMS: QMGR

Nazwa menedżera kolejek, z którym ma zostać nawiązane połączenie.

Domyślnie właściwość nie jest ustawiona.

XMSC_WMQ_RECEIVE_CCSID

Właściwość miejsca docelowego, która ustawia docelowy identyfikator CCSID dla konwersji komunikatów menedżera kolejek. Ta wartość jest ignorowana, chyba że właściwość XMSC_WMQ_RECEIVE_CONVERSION jest ustawiona na wartość WMQ_RECEIVE_CONVERSION_QMGR.

Typ danych:
Liczba całkowita

Wartość:
Dowolna dodatnia liczba całkowita.

Wartością domyślną jest 1208.

Określenie wartości `GMO_CONVERT` w komunikacie jest opcjonalne. Jeśli podana jest wartość `GMO_CONVERT` , konwersja odbywa się zgodnie z podaną wartością.

XMSC_WMQ_RECEIVE_CONVERSION

Właściwość miejsca docelowego, która określa, czy konwersja danych będzie wykonywana przez menedżer kolejek.

Typ danych:
Liczba całkowita

Wartości:
XMSC_WMQ_RECEIVE_CONVERSION_CLIENT_MSG (DEFAULT): konwersja danych jest wykonywana tylko na kliencie XMS . Konwersja jest zawsze wykonywana przy użyciu strony kodowej 1208.

XMSC_WMQ_RECEIVE_CONVERSION_QMGR: Konwersja danych w menedżerze kolejek przed wysłaniem komunikatu do klienta XMS .

XMSC_WMQ_RECEIVE_EXIT

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Identyfikuje wyjście odbierania kanału, które ma zostać uruchomione.

Wartością tej właściwości jest łańcuch, który identyfikuje wyjście odbierania kanału i ma następujący format:

libraryName(entryPointNazwa)

gdzie:

- **libraryName** to pełna ścieżka do zarządzanego wyjścia .dll
- entryPointNazwa jest nazwą klasy kwalifikowaną przez przestrzeń nazw

Na przykład C:\MyReceiveExit.dll(MyReceiveExitNameSpace.MyReceiveExitClassName)

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta zarządzanego. Obsługiwane są również tylko zarządzane wyjścia.

XMSC_WMQ_RECEIVE_EXIT_INIT

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Dane użytkownika przekazywane do wyjścia odbierania kanału w momencie jego wywołania.

Wartością właściwości jest łańcuch. Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta zarządzanego i jest ustawiona właściwość "XMSC_WMQ_RECEIVE_EXIT" na stronie 2142 .

XMSC_WMQ_RESOLVED_QUEUE_MANAGER

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Ta właściwość jest używana do uzyskiwania nazwy menedżera kolejek, z którym jest połączony.

W przypadku użycia z tabelą definicji kanału klienta (CCDT) ta nazwa może być inna niż nazwa menedżera kolejek określona w fabryce połączeń.

XMSC_WMQ_RESOLVED_QUEUE_MANAGER_ID

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Ta właściwość jest zapełniana identyfikatorem menedżera kolejek po nawiązaniu połączenia.

XMSC_WMQ_SECURITY_EXIT

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Identyfikuje wyjście zabezpieczeń kanału.

Wartością właściwości jest łańcuch, który identyfikuje wyjście zabezpieczeń kanału i ma następujący format:

libraryName(entryPointNazwa)

gdzie:

- **libraryName** to pełna ścieżka do pliku .dll zarządzanego wyjścia
- entryPointNazwa jest nazwą klasy kwalifikowaną przez przestrzeń nazw

Na przykład C:\MySecurityExit.dll(MySecurityExitNameSpace.MySecurityExitClassName)

Maksymalna długość łańcucha wynosi 128 znaków.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta zarządzanego. Obsługiwane są również tylko zarządzane wyjścia.

XMSC_WMQ_SECURITY_EXIT_INIT

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Dane użytkownika przekazywane do wyjścia zabezpieczeń kanału w momencie jego wywołania.

Maksymalna długość łańcucha danych użytkownika wynosi 32 znaki.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta zarządzanego i jest ustawiona właściwość “XMSC_WMQ_SECURITY_EXIT” na stronie 2143 .

XMSC_WMQ_SEND_EXIT

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Identyfikuje wyjście wysyłania kanału.

Wartością właściwości jest łańcuch. Wyjście wysyłania kanału ma następujący format:

libraryName(entryPointNazwa)

gdzie:

- **libraryName** to pełna ścieżka do pliku .dll zarządzanego wyjścia
- entryPointNazwa jest nazwą klasy kwalifikowaną przez przestrzeń nazw

Na przykład: C:\MySendExit.dll(MySendExitNameSpace.MySendExitClassName)

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta zarządzanego. Obsługiwane są również tylko zarządzane wyjścia.

XMSC_WMQ_SEND_EXIT_INIT

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Dane użytkownika przekazywane do wyjść wysyłania kanału w momencie ich wywołania.

Wartość właściwości jest łańcuchem zawierającym jeden lub więcej elementów danych użytkownika oddzielonych przecinkami. Domyślnie właściwość nie jest ustawiona.

Reguły określania danych użytkownika, które są przekazywane do sekwencji wyjść wysyłania kanału, są takie same, jak reguły określania danych użytkownika, które są przekazywane do sekwencji wyjść odbierania kanału. W związku z tym reguły można znaleźć w sekcji "XMSC_WMQ_RECEIVE_EXIT_INIT" na stronie 2143.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta zarządzanego i jest ustawiona właściwość "XMSC_WMQ_SEND_EXIT" na stronie 2144 .

XMSC_WMQ_SEND_CHECK_COUNT

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Liczba wywołań wysyłania dozwolonych między sprawdzeniami błędów asynchronicznego umieszczania w ramach jednej sesji XMS bez transakcji.

Domyślnie ta właściwość ma wartość 0.

XMSC_WMQ_SHARE_CONV_ALLOWED

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Odpowiednie obiekty:
Długa nazwa narzędzia administracyjnego JMS: SHARECONVALLOWED

Skrócona nazwa narzędzia administracyjnego JMS: SCALD

Określa, czy połączenie klienta może współużytkować gniazdo z innymi XMS połączeniami najwyższego poziomu z tego samego procesu do tego samego menedżera kolejek, jeśli definicje kanału są zgodne. Ta właściwość umożliwia pełną izolację połączeń w osobnych gniazdach (jeśli jest to konieczne z powodów związanych z tworzeniem, konserwacją lub działaniem aplikacji). Ustawienie tej właściwości wskazuje tylko wartość XMS , aby gniazdo bazowe było współużytkowane. Nie wskazuje ona, ile połączeń współużytkuje jedno gniazdo. Liczba połączeń współużytkujących gniazdo jest określana przez wartość SHARECNV, która jest negocjowana między klientem IBM MQ i serwerem IBM MQ .

Aplikacja może ustawić następujące stałe nazwane w celu ustawienia właściwości:

- XMSC_WMQ_SHARE_CONV_ALLOWED_FALSE-połączenia nie współużytkują gniazda.
- XMSC_WMQ_SHARE_CONV_ALLOWED_TRUE-Połączenia współużytkują gniazdo.

Domyślnie właściwość ma wartość XMSC_WMQ_SHARE_CONV_ALLOWED_ENABLED.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta.

XMSC_WMQ_SSL_CERT_STORES

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Lokalizacje serwerów, na których znajdują się listy odwołań certyfikatów (Certificate Revocation List – CRL) używane podczas nawiązywania połączenia SSL z menedżerem kolejek.

Wartością tej właściwości jest lista adresów URL rozdzielanych przecinkami. Każdy URL ma następujący format:

```
[user[/password]@]ldap://[serveraddress][:portnum][,...]
```

Ten format jest zgodny z podstawowym formatem MQJMS, ale jest od niego rozszerzony.

Poprawne jest puste pole `serveraddress`. W tym przypadku XMS przyjmuje, że wartością jest łańcuch "localhost".

Przykładowa lista:

```
myuser/mypassword@ldap://server1.mycom.com:389
ldap://server1.mycom.com
ldap://
ldap://:389
```

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Domyślnie właściwość nie jest ustawiona.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

XMSC_WMQ_SSL_CIPHER_SPEC

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa specyfikacji szyfrowania używanej w przypadku bezpiecznego połączenia z menedżerem kolejek.

W poniższej tabeli przedstawiono specyfikacje szyfrów, których można używać z obsługą protokołu TLS produktu IBM MQ . Jeśli żądasz certyfikatu osobistego, należy podać wielkość klucza dla pary kluczy publicznego i prywatnego. Wielkość klucza używanego podczas uzgadniania SSL jest wielkością zapisaną w certyfikacie, chyba że jest ona określona w specyfikacji szyfrowania CipherSpec(zgodnie z opisem w tabeli). Domyślnie ta właściwość nie jest ustawiona.

Nazwa specyfikacji szyfrowania	Używany protokół	Algorytm mieszający	Algorytm szyfrowania	Bity szyfrowania	FIPS[1]	Suite B 128-bitowy	Suite B 192-bitowy
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Tak	Nie	Nie
TLS_RSA_WITH_AES_256_CBC_SHA[2]	TLS 1.0	SHA-1	AES	256	Tak	Nie	Nie
TLS_RSA_WITH_DES_CBC_SHA	TLS 1.0	SHA-1	DES	56	Nie	Nie	Nie
TLS_RSA_WITH_3DES_EDE_CBC_SHA[4]	TLS 1.0	SHA-1	3DES	168	Tak	Nie	Nie
TLS_RSA_WITH_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Tak	Nie	Nie
TLS_RSA_WITH_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Tak	Nie	Nie

Nazwa specyfikacji szyfrowania	Używany protokół	Algorytm mieszający	Algorytm szyfrowania	Bity szyfrowania	FIPS[1]	Suite B 128-bitowy	Suite B 192-bitowy
TLS_RSA_WITH_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Tak	Nie	Nie
TLS_RSA_WITH_AES_256_CBC_SHA256	TLS 1.2	SHA-256	AES	256	Tak	Nie	Nie
ECDHE_ECDSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Nie	Nie	Nie
ECDHE_ECDSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Tak	Nie	Nie
ECDHE_RSA_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Nie	Nie	Nie
ECDHE_RSA_3DES_EDE_CBC_SHA256	TLS 1.2	SHA-256	3DES	168	Tak	Nie	Nie
ECDHE_ECDSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Tak	Nie	Nie
ECDHE_ECDSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Tak	Nie	Nie
ECDHE_RSA_AES_128_CBC_SHA256	TLS 1.2	SHA-256	AES	128	Tak	Nie	Nie
ECDHE_RSA_AES_256_CBC_SHA384	TLS 1.2	SHA-384	AES	256	Tak	Nie	Nie
ECDHE_ECDSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Tak	Tak	Nie
ECDHE_ECDSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Tak	Nie	Tak
ECDHE_RSA_AES_128_GCM_SHA256	TLS 1.2	SHA-256	AES	128	Tak	Nie	Nie
ECDHE_RSA_AES_256_GCM_SHA384	TLS 1.2	SHA-384	AES	256	Tak	Nie	Nie
TLS_RSA_WITH_NULL_SHA256	TLS 1.2	SHA-256	Brak	0	Nie	Nie	Nie
ECDHE_RSA_NULL_SHA256	TLS 1.2	SHA-256	Brak	0	Nie	Nie	Nie
ECDHE_ECDSA_NULL_SHA256	TLS 1.2	SHA-256	Brak	0	Nie	Nie	Nie
TLS_RSA_WITH_NULL_NULL	TLS 1.2	Brak	Brak	0	Nie	Nie	Nie
TLS_RSA_WITH_RC4_128_SHA256	TLS 1.2	SHA-256	RC4	128	Nie	Nie	Nie

Nazwa specyfikacji szyfrowania	Używany protokół	Algorytm mieszający	Algorytm szyfrowania	Bity szyfrowania	FIPS[1]	Suite B 128-bitowy	Suite B 192-bitowy

Uwagi:

1. Określa, czy specyfikacja szyfrowania CipherSpec jest zgodna ze standardami FIPS (Federal Information Processing Standards) 140-2. Wyjaśnienie standardu FIPS oraz informacje na temat konfigurowania produktu IBM MQ do obsługi operacji zgodnych ze standardem FIPS 140-2 zawiera sekcja FIPS (Federal Information Processing Standards).
2. Tej CipherSpec nie można użyć do zabezpieczenia połączenia produktu IBM MQ Explorer z menedżerem kolejek, chyba że odpowiednie nieograniczone pliki strategii zostaną zastosowane do środowiska JRE używanego przez produkt IBM MQ Explorer.
3. Ta specyfikacja szyfrowania uzyskała certyfikat FIPS 140-2 przed 19 maja 2007.
4. Po skonfigurowaniu programu IBM MQ dla operacji zgodnej ze standardem FIPS 140-2 ta specyfikacja szyfrowania może zostać użyta do przesłania maksymalnie 32 GB danych. Po przekroczeniu tej wartości połączenie zostanie przerwane i zostanie wyświetlony komunikat o błędzie AMQ9288. Aby uniknąć tego błędu, należy unikać używania potrójnego algorytmu szyfrowania DES (który jest nieaktualny) lub włączyć resetowanie klucza tajnego w przypadku używania tej CipherSpec w konfiguracji FIPS 140-2.

Pojęcia pokrewne

Integralność danych komunikatów

Zadania pokrewne

Zabezpieczanie

Określanie CipherSpecs

XMSC_WMQ_SSL_CIPHER_SUITE

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa zestawu algorytmów szyfrowania używanego w przypadku połączenia TLS z menedżerem kolejek. Na podstawie podanego zestawu algorytmów szyfrowania jest określany protokół używany do negocjowania bezpiecznego połączenia.

Ta właściwość ma następujące wartości kanoniczne:

- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA
- SSL_RSA_EXPORT1024_WITH_RC4_56_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- SSL_RSA_WITH_AES_256_CBC_SHA
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA

Tę wartość można podać zamiast wartości XMSC_WMQ_SSL_CIPHER_SPEC.

Jeśli dla właściwości XMSC_WMQ_SSL_CIPHER_SPECokreślono wartość niepustą, ta wartość nadpisuje ustawienie właściwości XMSC_WMQ_SSL_CIPHER_SUITE. Jeśli zmienna XMSC_WMQ_SSL_CIPHER_SPEC nie ma wartości, jako zestaw algorytmów szyfrowania dla produktu IBM Global Security Kit (GSKit)jest używana wartość zmiennej XMSC_WMQ_SSL_CIPHER_SUITE. W tym przypadku wartość jest odwzorowywana na równoważną wartość CipherSpec , zgodnie z opisem w sekcji CipherSuite i CipherSpec dla odwzorowań nazw na potrzeby połączeń XMS z menedżerem kolejek produktu IBM MQ.

Jeśli obie wartości XMSC_WMQ_SSL_CIPHER_SPEC i XMSC_WMQ_SSL_CIPHER_SUITE są puste, pole pChDef->SSLCipherSpec jest wypełnione spacjami.

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Domyślnie właściwość nie jest ustawiona.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

XMSC_WMQ_SSL_CRYPTO_HW

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Szczegóły konfiguracji sprzętu szyfrującego połączonego z systemem klienta.

Ta właściwość ma następujące wartości kanoniczne:

- GSK_ACCELERATOR_RAINBOW_CS_OFF
- GSK_ACCELERATOR_RAINBOW_CS_ON
- GSK_ACCELERATOR_NCIPHER_NF_OFF
- GSK_ACCELERATOR_NCIPHER_NF_ON

Istnieje specjalny format dla sprzętu szyfrującego PKCS11 (gdzie DriverPath, TokenLabeli TokenPassword są łańcuchami podanymi przez użytkownika):

```
GSK_PKCS11=PKCS#11 DriverPath; PKCS#11 TokenLabel;PKCS#11 TokenPassword
```

XMS nie interpretuje ani nie zmienia treści łańcucha. Kopiowana jest podana wartość (maksymalnie 256 znaków jednobajtowych) do obiektu MQSCO MQSCO.CryptoHardware .

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Domyślnie właściwość nie jest ustawiona.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

XMSC_WMQ_SSL_FIPS_REQUIRED

Typ danych:
Boolowski

Właściwość:
ConnectionFactory

Wartość tej właściwości określa, czy aplikacja może lub nie może używać zestawów algorytmów szyfrowania niezgodnych ze standardem FIPS. Jeśli ta właściwość zostanie ustawiona na wartość true, w przypadku połączeń klient-serwer będzie można używać tylko algorytmów zgodnych ze standardem FIPS.

Ta właściwość może mieć następujące wartości, które są tłumaczone na dwie wartości kanoniczne dla MQSCO.FipsRequired:

Tabela 880. Tabela wartości parametru MQSCO.FlipsRequired

Wartość	**Opis**	**Odpowiednia wartość parametru MQSCO.FipsRequired**
Fałsz	Można użyć dowolnej CipherSpec .	MQSSL_FIPS_NO (wartość domyślna)
Prawda	W specyfikacji szyfrowania CipherSpec mającej zastosowanie do tego połączenia klienta mogą być używane tylko algorytmy szyfrowania z certyfikatem FIPS.	MQSSL_FIPS_YES

XMS kopiuje odpowiednią wartość do MQSCO.FipsRequired przed wywołaniem MQCONNX.

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

XMSC_WMQ_SSL_KEY_REPOSITORY

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Położenie pliku bazy danych kluczy, w którym przechowywane są klucze i certyfikaty.

XMS kopiuje łańcuch, maksymalnie 256 znaków jednobajtowych, do obiektu MQSCO MQSCO.KeyRepository . IBM MQ interpretuje ten łańcuch jako nazwę pliku wraz z pełną ścieżką.

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Domyślnie właściwość nie jest ustawiona.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

XMSC_WMQ_SSL_KEY_RESETCOUNT,

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Wartość KeyResetCount reprezentuje całkowitą liczbę nieszyfrowanych bajtów wysłanych i odebranych w ramach konwersacji SSL przed ponownym wynegocjowaniem klucza tajnego. Liczba bajtów obejmuje informacje sterujące wysłane przez agenta MCA.

XMS kopiuje wartość podaną dla tej właściwości do MQSCO.KeyResetCount przed wywołaniem MQCONNX.

Parametr MQSCO.KeyRestCount jest dostępna tylko w produkcie IBM MQ w wersji 6. W przypadku systemu IBM MQ w wersji 5.3, jeśli ta właściwość jest ustawiona, system XMS nie podejmuje próby nawiązania połączenia z menedżerem kolejek i zamiast tego zgłasza odpowiedni wyjątek.

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Wartością domyślną tej właściwości jest zero, co oznacza, że klucze tajne nigdy nie są renegocjowane.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

XMSC_WMQ_SSL_PEER_NAME

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa węzła sieci używana na potrzeby połączenia SSL z menedżerem kolejek.

Dla tej właściwości nie ma listy wartości kanonicznych. Zamiast tego należy zbudować ten łańcuch zgodnie z regułami dotyczącymi komendy SSLPEER.

Przykładowa nazwa węzła sieci:

```
"CN=John Smith, O=IBM ,OU=Test , C=GB"
```

XMS kopiuje łańcuch do poprawnej jednobajtowej strony kodowej i umieszcza poprawne wartości w polach MQCD.SSLPeerNamePtr i MQCD.SSLPeerNameLength przed wywołaniem MQCONNX.

Ta właściwość ma zastosowanie tylko wtedy, gdy aplikacja nawiązuje połączenie z menedżerem kolejek w trybie klienta.

Tylko w przypadku systemu .NET : zarówno połączenia zarządzane z produktem IBM MQ 8.0do produktu IBM MQ (WMQ_CM_CLIENT), jak i połączenia niezarządzane z produktem IBM MQ (WMQ_CM_CLIENT_UNMANAGED) obsługują połączenia TLS/SSL.

Domyślnie właściwość nie jest ustawiona.

Pojęcia pokrewne

Obsługa protokołów SSL i TLS dla niezarządzanego klienta .NET

Obsługa SSL i TLS dla zarządzanego klienta .NET

Odsyłacze pokrewne

SSLPEERNAME

XMSC_WMQ_SYNCPOINT_ALL_GETS

Typ danych:
System.Boolean

Właściwość:
ConnectionFactory

Określa, czy wszystkie komunikaty muszą być pobierane z kolejek w ramach sterowania punktem synchronizacji.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
Fałsz	Jeśli okoliczności są odpowiednie, klient XMS może pobierać komunikaty z kolejek poza kontrolą punktu synchronizacji.
Prawda	Klient XMS musi pobrać wszystkie komunikaty z kolejek w ramach kontroli punktu synchronizacji.

Wartością domyślną jest false.

XMSC_WMQ_TARGET_CLIENT

Typ danych:
System.Int32

Właściwość:
Miejsce docelowe

Nazwa używana w identyfikatorze URI:
targetClient

Określa, czy komunikaty wysyłane do miejsca docelowego mają zawierać nagłówek MQRFH2.

Jeśli aplikacja wysyła komunikat zawierający nagłówek MQRFH2 , aplikacja odbierająca musi być w stanie obsłużyć ten nagłówek.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WMQ_TARGET_DEST_JMS	Komunikaty wysyłane do miejsca docelowego zawierają nagłówek MQRFH2 . Tę wartość należy określić, jeśli aplikacja wysyła komunikaty do innej aplikacji XMS , aplikacji IBM MQ classes for JMS lub rodzimej aplikacji IBM MQ , która została zaprojektowana do obsługi nagłówka MQRFH2 .
XMSC_WMQ_TARGET_DEST_MQ	Komunikaty wysyłane do miejsca docelowego nie zawierają nagłówka MQRFH2 . Tę wartość należy określić, jeśli aplikacja wysyła komunikaty do rodzimej aplikacji IBM MQ , która nie jest zaprojektowana do obsługi nagłówka MQRFH2 .

Wartością domyślną jest XMSC_WMQ_TARGET_DEST_JMS.

XMSC_WMQ_TEMP_Q_PREFIX

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Przedrostek używany do tworzenia nazwy kolejki dynamicznej IBM MQ , która jest tworzona podczas tworzenia przez aplikację kolejki tymczasowej XMS .

Reguły tworzenia przedrostka są takie same, jak reguły tworzenia zawartości pola **DynamicQName** w deskryptorze obiektu, ale ostatni niepusty znak musi być znakiem gwiazdki (*). Jeśli ta właściwość nie jest ustawiona, używana jest wartość `CSQ.*` w systemie z/OS i `AMQ.*` na innych platformach. Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko w domenie typu punkt z punktem.

XMSC_WMQ_TEMP_TOPIC_PREFIX

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory, Miejsce docelowe

Podczas tworzenia tematów tymczasowych produkt XMS generuje łańcuch tematu w postaci "TEMP/TEMPTOPICPREFIX/unique_id" lub, jeśli ta właściwość zawiera wartość domyślną, generowany jest łańcuch "TEMP/unique_id". Podanie niepustej wartości umożliwia definiowanie konkretnych kolejek modelowych na potrzeby tworzenia zarządzanych kolejek dla subskrybentów tymczasowych tematów utworzonych w ramach tego połączenia.

Każdy łańcuch o wartości innej niż NULL, składający się tylko z poprawnych znaków dla łańcucha tematu produktu IBM MQ , jest poprawną wartością tej właściwości.

Domyślnie ta właściwość jest ustawiona na wartość "" (pusty łańcuch).

Uwaga: Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WMQ_TEMPORARY_MODEL

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa kolejki modelowej IBM MQ , na podstawie której tworzona jest kolejka dynamiczna, gdy aplikacja tworzy kolejkę tymczasową XMS .

Wartością domyślną tej właściwości jest SYSTEM.DEFAULT.MODEL.QUEUE.

Ta właściwość ma zastosowanie tylko w domenie typu punkt z punktem.

XMSC_WMQ_WILDCARD_FORMAT

Typ danych:
System.Int32

Właściwość:
ConnectionFactory, Miejsce docelowe

Ta właściwość określa, która wersja składni znaku wieloznacznego ma być używana.

W przypadku korzystania z publikowania/subskrybowania z użyciem znaków IBM MQ '*' i '?' są traktowane jako znaki wieloznaczne. Natomiast znaki '#' i '+' są traktowane jako znaki wieloznaczne w przypadku korzystania z subskrypcji publikowania w produkcie IBM Integration Bus. Ta właściwość zastępuje właściwość XMSC_WMQ_BROKER_VERSION.

Poprawne wartości dla tej właściwości to:

XMSC_WMQ_WILDCARD_TOPIC_ONLY,
Rozpoznaje tylko znaki wieloznaczne na poziomie tematu, np. "#" i '+' są traktowane jako znaki wieloznaczne. Ta wartość jest taka sama jak XMSC_WMQ_BROKER_V2.

XMSC_WMQ_WILDCARD_CHAR_ONLY
Rozpoznaje tylko znaki wieloznaczne, np. "*" i "?" są traktowane jako znaki wieloznaczne. Ta wartość jest taka sama jak XMSC_WMQ_BROKER_V1.

Domyślnie ta właściwość ma wartość XMSC_WMQ_WILDCARD_TOPIC_ONLY.

XMSC_WPM_BUS_NAME

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory i Destination

Nazwa używana w identyfikatorze URI:
busName

W przypadku fabryki połączeń: nazwa magistrali integracji usług, z którą aplikacja nawiązuje połączenie. W przypadku miejsca docelowego: nazwa magistrali integracji usług, w której znajduje się miejsce docelowe.

W przypadku miejsca docelowego będącego tematem ta właściwość jest nazwą magistrali integracji usług, w której istnieje powiązany obszar tematu. Ten obszar tematu jest określony przez właściwość XMSC_WPM_TOPIC_SPACE .

Jeśli właściwość nie jest ustawiona dla miejsca docelowego, przyjmuje się, że kolejka lub powiązany obszar tematu istnieją na magistrali integracji usług, z którą łączy się aplikacja.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_CONNECTION_PROTOCOL

Typ danych:
System.Int32

Właściwość:
Połączenie

Protokół komunikacyjny używany na potrzeby połączenia z mechanizmem przesyłania komunikatów. Ta właściwość jest tylko do odczytu.

Możliwe wartości tej właściwości są następujące:

Wartość	Znaczenie
XMSC_WPM_CP_HTTP	Połączenie korzysta z protokołu HTTP przez TCP/IP.
XMSC_WPM_CP_TCP	Połączenie używa protokołu TCP/IP.

XMSC_WPM_CONNECTION_BLISKOŚĆ

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Ustawienie bliskości połączeń na potrzeby nawiązywania połączenia. Ta właściwość określa, jak blisko musi być mechanizm przesyłania komunikatów, z którym aplikacja nawiązuje połączenie, z serwerem startowym.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Ustawienie bliskości połączenia
XMSC_WPM_CONNECTION_PROXIMITY_BUS	Magistrala
XMSC_WPM_CONNECTION_PROXIMITY_CLUSTER	Klaster
XMSC_WPM_CONNECTION_PROXIMITY_HOST	Host
XMSC_WPM_CONNECTION_PROXIMITY_SERVER	Serwer

Wartością domyślną jest XMSC_WPM_CONNECTION_PROXIMITY_BUS.

XMSC_WPM_DUR_SUB_HOME (podkatalog)

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa używana w identyfikatorze URI:
durableSubscription-strona główna

Nazwa mechanizmu przesyłania komunikatów zarządzającego wszystkimi trwałymi subskrypcjami połączeń lub miejsc docelowych. Komunikaty dostarczane do trwałych subskrybentów są przechowywane w punkcie publikacji tego samego mechanizmu przesyłania komunikatów.

Aby aplikacja mogła utworzyć trwały subskrybent korzystający z połączenia, należy określić punkt subskrypcji trwałej dla połączenia. Każda wartość określona dla miejsca docelowego nadpisuje wartość określoną dla połączenia.

Domyślnie właściwość nie jest ustawiona.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WPM_HOST_NAME

Typ danych:
Łańcuch

Właściwość:
Połączenie

Położenie mechanizmu przesyłania komunikatów, z którym połączona jest aplikacja (nazwa hosta lub adres IP systemu). Ta właściwość jest tylko do odczytu.

XMSC_WPM_LOCAL_ADDRESS

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

W przypadku połączenia z magistralą integracji usług: ta właściwość określa używaną wartość interfejsu sieci lokalnej albo portu lokalnego (lub zakresu portów lokalnych) bądź obie te wartości.

Wartością właściwości jest łańcuch w następującym formacie:

[*nazwa_hosta*] [(*low_port*) [,*port_wysoki*])]

Znaczenie zmiennych jest następujące:

nazwa_hosta
Nazwa hosta lub adres IP lokalnego interfejsu sieciowego, który ma być używany dla połączenia.

Podanie tych informacji jest konieczne tylko wtedy, gdy system, w którym działa aplikacja, ma dwa lub więcej interfejsów sieciowych, a użytkownik musi mieć możliwość określenia interfejsu, który ma być używany dla połączenia. Jeśli system ma tylko jeden interfejs sieciowy, można użyć tylko tego interfejsu. Jeśli system ma dwa lub więcej interfejsów sieciowych i nie określono interfejsu, który ma być używany, interfejs jest wybierany losowo.

low_port (low_port)
Numer portu lokalnego, który ma być używany dla połączenia.

Jeśli określono również wartość *high_port* , wartość *low_port* jest interpretowana jako najniższy numer portu z zakresu numerów portów.

port_wysoki
Najwyższy numer portu w zakresie numerów portów. Dla połączenia należy użyć jednego z portów z podanego zakresu.

Poniżej przedstawiono przykłady poprawnych wartości właściwości:

JUPITER
9.20.4.98
JUPITER (1000)

9.20.4.98(1000,2000)
(1000)
(1000,2000)

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_ME_NAME

Typ danych:
Łańcuch

Właściwość:
Połączenie

Nazwa mechanizmu przesyłania komunikatów, z którym połączona jest aplikacja. Ta właściwość jest tylko do odczytu.

XMSC_WPM_NON_PERSISTENT_MAP

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Poziom niezawodności komunikatów nietrwałych wysyłanych za pośrednictwem połączenia.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Poziom niezawodności
XMSC_WPM_MAPPING_AS_DESTINATION	Określany przez domyślny poziom niezawodności określony dla kolejki lub obszaru tematu w magistrali integracji usług.
XMSC_WPM_MAPPING_BEST_EFFORT_NON_ Trwały	Możliwie optymalny nietrwały
XMSC_WPM_MAPPING_EXPRESS_NON_ Trwały	Ekspresowy nietrwały
XMSC_WPM_MAPPING_RELIABLE_NON_ Trwały	Niezawodny nietrwały
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Niezawodny trwały
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Gwarantowany trwały

Wartością domyślną jest XMSC_WPM_MAPPING_EXPRESS_NON_PERSISTENT.

XMSC_WPM_PERSISTENT_MAP

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Poziom niezawodności komunikatów trwałych wysyłanych za pośrednictwem połączenia.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Poziom niezawodności
XMSC_WPM_MAPPING_AS_DESTINATION	Określany przez domyślny poziom niezawodności określony dla kolejki lub obszaru tematu w magistrali integracji usług.
XMSC_WPM_MAPPING_BEST_EFFORT_NON_Trwały	Możliwie optymalny nietrwały
XMSC_WPM_MAPPING_EXPRESS_NON_Trwały	Ekspresowy nietrwały
XMSC_WPM_MAPPING_RELIABLE_NON_Trwały	Niezawodny nietrwały
XMSC_WPM_MAPPING_RELIABLE_PERSISTENT	Niezawodny trwały
XMSC_WPM_MAPPING_ASSURED_PERSISTENT	Gwarantowany trwały

Wartością domyślną jest XMSC_WPM_MAPPING_RELIABLE_PERSISTENT.

XMSC_WPM_PORT

Typ danych:
System.Int32

Właściwość:
Połączenie

Numer portu, na którym nasłuchuje mechanizm przesyłania komunikatów, z którym aplikacja nawiązała połączenie. Ta właściwość jest tylko do odczytu.

Punkty końcowe dostawcy XMSC_WPM_PROVIDER_ENDPOINTS

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Jeden adres punktu końcowego serwera startowego lub sekwencja wielu adresów. Adresy punktów końcowych są rozdzielane przecinkami.

Serwer startowy to serwer aplikacji odpowiedzialny za wybór mechanizmu przesyłania komunikatów, z którym łączy się aplikacja. Adres punktu końcowego serwera startowego ma następujący format:

nazwa_hosta:*numer_portu*:*nazwa_łańcucha*

Znaczenia komponentów adresu punktu końcowego są następujące:

nazwa_hosta
Nazwa hosta lub adres IP systemu, w którym znajduje się serwer startowy. Jeśli nie określono nazwy hosta ani adresu IP, wartością domyślną jest `localhost`.

numer_portu
Numer portu, na którym serwer startowy nasłuchuje żądań przychodzących. Jeśli nie określono numeru portu, wartością domyślną jest 7276.

nazwa_łańcucha
Nazwa startowego łańcucha transportowego używanego przez serwer startowy. Poprawne wartości to:

Poprawna wartość	Nazwa startowego łańcucha transportowego
XMSC_WPM_BOOTSTRAP_HTTP	Przesyłanie komunikatów BootstrapTunneled
XMSC_WPM_BOOTSTRAP_HTTPS	BootstrapTunneledSecureMessaging
XMSC_WPM_BOOTSTRAP_SSL	Przesyłanie komunikatów BootstrapSecure
XMSC_WPM_BOOTSTRAP_TCP	Przesyłanie komunikatów BootstrapBasic

Jeśli nie zostanie podana żadna nazwa, wartością domyślną jest XMSC_WPM_BOOTSTRAP_TCP.

Jeśli nie określono adresu punktu końcowego, wartością domyślną jest localhost:7276:BootstrapBasicMessaging.

XMSC_WPM_SSL_CIPHER_SUITE

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa zestawu algorytmów szyfrowania CipherSuite , który ma być używany w połączeniu TLS z mechanizmem przesyłania komunikatów produktu WebSphere Application Server service integration bus . Na podstawie podanego zestawu algorytmów szyfrowania jest określany protokół używany do negocjowania bezpiecznego połączenia.

Tabela 881. Opcje pakietu CipherSuite na potrzeby połączenia z mechanizmem przesyłania komunikatów produktu WebSphere Application Server service integration bus

zestaw algorytmów szyfrowania	Używany protokół
TLS_RSA_WITH_DES_CBC_SHA	TLSv1
TLS_RSA_WITH_3DES_EDE_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_128_CBC_SHA	TLSv1
TLS_RSA_WITH_AES_256_CBC_SHA	TLSv1

Uwagi:

1. Windows TLS_RSA_WITH_AES_128_CBC_SHA i TLS_RSA_WITH_AES_256_CBC_SHA CipherSuites są obsługiwane tylko w systemie Windows . (Jest to podyktowane przez GSKit).
2. Deprecated TLS_RSA_WITH_3DES_EDE_CBC_SHA jest nieaktualna. Jednak nadal może być używany do przesyłania do 32 GB danych, zanim połączenie zostanie przerwane i zostanie zgłoszony błąd AMQ9288. Aby uniknąć tego błędu, należy unikać używania potrójnego algorytmu szyfrowania DES lub włączyć resetowanie klucza tajnego podczas korzystania z tej CipherSpec.

Dla tej właściwości nie ma wartości domyślnej. Aby używać protokołu SSL lub TLS, należy określić wartość tej właściwości. W przeciwnym razie aplikacja nie będzie mogła pomyślnie nawiązać połączenia z serwerem.

XMSC_WPM_SSL_FIPS_REQUIRED

Uwaga: W systemie AIX, Linux, and Windows IBM MQ zapewnia zgodność ze standardem FIPS 140-2 za pośrednictwem modułu szyfrującego IBM Crypto for C (ICC) . Certyfikat dla tego modułu został przeniesiony do statusu historycznego. Klienci powinni zapoznać się z informacjami w sekcji Certyfikat IBM Crypto for C (ICC) i zapoznać się z poradami NIST. Zastępczy moduł FIPS 140-3 jest obecnie w toku, a jego status można wyświetlić, wyszukując go na liście Moduły NIST CMVP na liście procesów.

Typ danych:
Boolowski

Właściwość:
ConnectionFactory

Wartość tej właściwości określa, czy aplikacja może lub nie może używać zestawów algorytmów szyfrowania niezgodnych ze standardem FIPS. Jeśli ta właściwość jest ustawiona na wartość true, dla połączenia klient-serwer używane są tylko algorytmy FIPS.Ustawienie wartości tej właściwości na TRUE uniemożliwia aplikacji korzystanie z zestawów algorytmów szyfrowania niezgodnych ze standardem FIPS.

Domyślnie właściwość ma wartość FALSE (czyli tryb FIPS jest wyłączony).

XMSC_WPM_SSL_KEY_REPOSITORY (repozytorium kluczy SSL)

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Ścieżka do pliku kluczy zawierającego klucze publiczne lub prywatne, który ma być używany w połączeniu chronionym.

Ustawienie właściwości pliku kluczy na wartość specjalną XMSC_WPM_SSL_MS_CERTIFICATE_STORE określa użycie bazy danych kluczy Microsoft Windows . Korzystanie z bazy danych kluczy Microsoft Windows , która znajduje się w panelu **Control Panel** > **Opcje internetowe** > **Zawartość** > **Certyfikaty**, eliminuje potrzebę korzystania z oddzielnej bazy danych plików kluczy. Użycie tej stałej w systemie Windows x64 i na innych platformach jest niedozwolone.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_SSL_KEYRING_LABEL

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Certyfikat używany podczas uwierzytelniania na serwerze. Jeśli nie zostanie podana żadna wartość, zostanie użyty certyfikat domyślny.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_SSL_KEYRING_PW,

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Hasło do pliku kluczy.

Tej właściwości można użyć jako alternatywy dla używania komendy XMSC_WPM_SSL_KEYRING_STASH_FILE do skonfigurowania hasła dla pliku kluczy.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_SSL_KEYRING_STASH_FILE (XMSC_WPM_KEYRING_STASH_FILE)

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa pliku binarnego zawierającego hasło do pliku repozytorium kluczy.

Tej właściwości można użyć jako alternatywy dla używania komendy XMSC_WPM_SSL_KEYRING_PW do konfigurowania hasła dla pliku kluczy.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_TARGET_GROUP

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa grupy docelowej mechanizmów przesyłania komunikatów. Charakter grupy docelowej jest określany przez właściwość XMSC_WPM_TARGET_TYPE .

Tę właściwość należy ustawić, aby ograniczyć wyszukiwanie mechanizmu przesyłania komunikatów do podgrupy mechanizmów przesyłania komunikatów na magistrali integracji usług. Jeśli aplikacja ma mieć możliwość nawiązania połączenia z dowolnym mechanizmem przesyłania komunikatów na magistrali integracji usług, nie należy ustawiać tej właściwości.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_TARGET_ISTOTNOŚĆ

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Znaczenie grupy docelowej mechanizmów przesyłania komunikatów.

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WPM_TARGET_ISTOTNOŚCI Preferowane	Mechanizm przesyłania komunikatów w grupie docelowej jest wybierany, jeśli jest dostępny. W przeciwnym razie zostanie wybrany mechanizm przesyłania komunikatów spoza grupy docelowej, pod warunkiem, że znajduje się on w tej samej magistrali integracji usług.
XMSC_WPM_TARGET_ISTOTNOŚCI WYMAGANE	Wybrany mechanizm przesyłania komunikatów musi należeć do grupy docelowej. Jeśli mechanizm przesyłania komunikatów w grupie docelowej jest niedostępny, proces połączenia kończy się niepowodzeniem.

Wartością domyślną tej właściwości jest XMSC_WPM_TARGET_IMPORTANT.

XMSC_WPM_TARGET_TRANSPORT_CHAIN

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Nazwa łańcucha transportowego danych przychodzących, który musi być używany przez aplikację do nawiązywania połączenia z mechanizmem przesyłania komunikatów.

Wartością tej właściwości może być nazwa dowolnego łańcucha transportowego danych przychodzących, który jest dostępny na serwerze aplikacji udostępniającym mechanizm przesyłania komunikatów. Dla

jednego z predefiniowanych łańcuchów transportu przychodzącego udostępniono następującą stałą nazwaną:

Stała nazwana	Nazwa łańcucha transportowego
XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC	Przesyłanie komunikatów InboundBasic

Wartością domyślną tej właściwości jest XMSC_WPM_TARGET_TRANSPORT_CHAIN_BASIC.

XMSC_WPM_TARGET_TYPE

Typ danych:
System.Int32

Właściwość:
ConnectionFactory

Typ grupy docelowej mechanizmów przesyłania komunikatów. Ta właściwość określa rodzaj grupy docelowej identyfikowanej przez właściwość XMSC_WPM_TARGET_GROUP .

Poprawne wartości właściwości są następujące:

Poprawna wartość	Znaczenie
XMSC_WPM_TARGET_TYPE_BUSMEMBER	Nazwa grupy docelowej jest nazwą elementu magistrali. Grupą docelową są wszystkie mechanizmy przesyłania komunikatów w elemencie magistrali.
XMSC_WPM_TARGET_TYPE_CUSTOM	Nazwa grupy docelowej jest nazwą zdefiniowanej przez użytkownika grupy mechanizmów przesyłania komunikatów. Grupa docelowa to wszystkie mechanizmy przesyłania komunikatów zarejestrowane w grupie zdefiniowanej przez użytkownika.
XMSC_WPM_TARGET_TYPE_ME	Nazwa grupy docelowej jest nazwą mechanizmu przesyłania komunikatów. Grupa docelowa jest określonym mechanizmem przesyłania komunikatów.

Domyślnie właściwość nie jest ustawiona.

XMSC_WPM_TEMP_Q_PREFIX

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Przedrostek używany do tworzenia nazwy kolejki tymczasowej, która jest tworzona w magistrali integracji usług podczas tworzenia przez aplikację kolejki tymczasowej XMS . Przedrostek może zawierać maksymalnie 12 znaków.

Nazwa kolejki tymczasowej rozpoczyna się od znaków "_Q", po których następuje przedrostek. Pozostała część nazwy składa się ze znaków wygenerowanych przez system.

Domyślnie właściwość nie jest ustawiona, co oznacza, że nazwa kolejki tymczasowej nie ma przedrostka.

Ta właściwość ma zastosowanie tylko w domenie typu punkt z punktem.

XMSC_WPM_TEMP_TOPIC_PREFIX

Typ danych:
Łańcuch

Właściwość:
ConnectionFactory

Przedrostek używany do generowania nazwy tematu tymczasowego tworzonego przez aplikację. Przedrostek może zawierać maksymalnie 12 znaków.

Nazwa tematu tymczasowego rozpoczyna się od znaków "_T", po których następuje przedrostek. Pozostała część nazwy składa się ze znaków wygenerowanych przez system.

Domyślnie właściwość nie jest ustawiona, co oznacza, że nazwa tematu tymczasowego nie ma przedrostka.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

XMSC_WPM_TOPIC_SPACE

Typ danych:
Łańcuch

Właściwość:
Miejsce docelowe

Nazwa używana w identyfikatorze URI:
topicSpace

Nazwa obszaru tematu zawierającego dany temat. Ta właściwość może mieć tylko miejsce docelowe, które jest tematem.

Domyślnie właściwość nie jest ustawiona, co oznacza, że przyjmowany jest domyślny obszar tematu.

Ta właściwość ma zastosowanie tylko w domenie publikowania/subskrypcji.

Informacje dodatkowe o programowaniu aplikacji w systemie Managed File Transfer

Informacje uzupełniające ułatwiające tworzenie aplikacji dla produktu Managed File Transfer.

Przykłady używania komendy fteCreateTransfer do uruchamiania programów

Komenda **fteCreateTransfer** umożliwia określenie programów, które mają być uruchomione przed lub po przesłaniu.

Oprócz korzystania z programu **fteCreateTransfer**istnieją inne sposoby wywoływania programu przed przesłaniem lub po nim. Więcej informacji na ten temat zawiera sekcja Określanie programów do uruchomienia z produktem MFT.

We wszystkich tych przykładach do określenia programu używana jest następująca składnia:

```
[type:]commandspec[,[retrycount][,[retrywait][,successrc]]]
```

Więcej informacji na temat tej składni zawiera sekcja **fteCreateTransfer**: rozpoczynanie nowego przesyłania plików.

Uruchamianie programu wykonywalnego

Poniższy przykład określa program wykonywalny o nazwie `mycommand` i przekazuje do programu dwa argumenty, a i b.

```
mycommand(a,b)
```

Aby uruchomić ten program na agencie źródłowym AGENT1 przed rozpoczęciem przesyłania, należy użyć następującej komendy:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

Uruchamianie i ponawianie programu wykonywalnego

W poniższym przykładzie określono program wykonywalny o nazwie `simple`, który nie ma żadnych argumentów. Wartość 1 jest określana dla parametru `retrycount` , a wartość 5 dla parametru `retrywait`. Te wartości oznaczają, że program zostanie ponowiony raz, jeśli nie zwróci kodu powrotu po upływie pięciu sekund. Nie podano żadnej wartości dla `successrc`, więc jedynym pomyślnym kodem powrotu jest wartość domyślna 0.

```
executable:simple,1,5
```

Aby uruchomić ten program na agencie źródłowym AGENT1 po zakończeniu przesyłania, należy użyć następującej komendy:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Uruchamianie skryptu Ant i określanie pomyślnych kodów powrotu

Poniższy przykład określa skrypt Ant o nazwie `myscript` i przekazuje do niego dwie właściwości. Skrypt jest uruchamiany za pomocą komendy **`fteAnt`** . Wartość `successrc` jest określana jako >2&<7&!5|0|14, co oznacza, że kody powrotu 0, 3, 4, 6 i 14 oznaczają powodzenie.

```
antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14
```

Aby uruchomić ten program na agencie docelowym AGENT2 przed rozpoczęciem przesyłania, należy użyć następującej komendy:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst
"antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14"destinationSpecification sourceSpecification
```

Uruchamianie skryptu Ant i określanie obiektów docelowych do wywołania

W poniższym przykładzie określono skrypt Ant o nazwie `script2` i dwa obiekty docelowe, `target1` i `target2`, do wywołania. Przekazywana jest również właściwość `prop1` o wartości `recmfm(F,B)`. Nawiasy, przecinki (,) i ukośniki odwrotne (\) są znakami specjalnymi w komendach MFT i muszą być poprzedzone znakiem ukośnika odwrotnego (\). Windows Ścieżki do plików w systemie Windows można określić przy użyciu podwójnych ukośników odwrotnych (\\) jako separatora lub przy użyciu pojedynczych ukośników (/). W poniższym przykładzie przecinek (,) i nawiasy są poprzedzone znakiem ukośnika odwrotnego (\).

```
antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14
```

Aby uruchomić ten program w agencie docelowym AGENT2 po zakończeniu przesyłania, należy użyć następującej komendy:

```
fteCreateTransfer -sa AGENT1 -da AGENT2
-postdst "antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14"
destinationSpecification sourceSpecification
```

Korzystanie z metadanych w skrypcie Ant

Zadanie Ant można określić jako dowolne z następujących wywołań przesyłania:

- Przed, źródło
- Po, źródło
- wstępne miejsce docelowe
- miejsce docelowe po

Po uruchomieniu zadania Ant metadane użytkownika operacji przesyłania są udostępniane przy użyciu zmiennych środowiskowych. Dostęp do tych danych można uzyskać na przykład za pomocą następującego kodu:

```
<property environment="environment" />
<echo>${environment.mymetadata}</echo>
```

gdzie `mymetadata` jest nazwą metadanych wstawionych do operacji przesyłania.

Uruchamianie skryptu JCL

W poniższym przykładzie określono skrypt JCL o nazwie Z0SBATCH. Dla parametru `retrycount`określono wartość 3, dla parametru `retrywait` określono wartość 30, a dla parametru `successrc`określono wartość 0. Te wartości oznaczają, że skrypt jest ponawiany trzy razy, jeśli nie zwróci kodu powrotu 0, z trzydziestosekundowym czasem oczekiwania między kolejnymi próbami.

```
jcl:ZOSBATCH,3,30,0
```

gdzie ZOSBATCH jest elementem zestawu PDS o nazwie MYSYS.JCL, a plik `agent.properties` zawiera wiersz `commandPath=....://'MYSYS.JCL':...`

Aby uruchomić ten program na agencie źródłowym AGENT1 po zakończeniu przesyłania, należy użyć następującej komendy:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:ZOSBATCH,3,30,0
destinationSpecification sourceSpecification
```

Zadania pokrewne

Określanie programów, które mają być uruchamiane z systemem MFT

Odsyłacze pokrewne

`fteCreateTransfer`: rozpoczęcie nowego przesyłania plików

`fteAnt`: uruchamianie zadań Ant w programie MFT

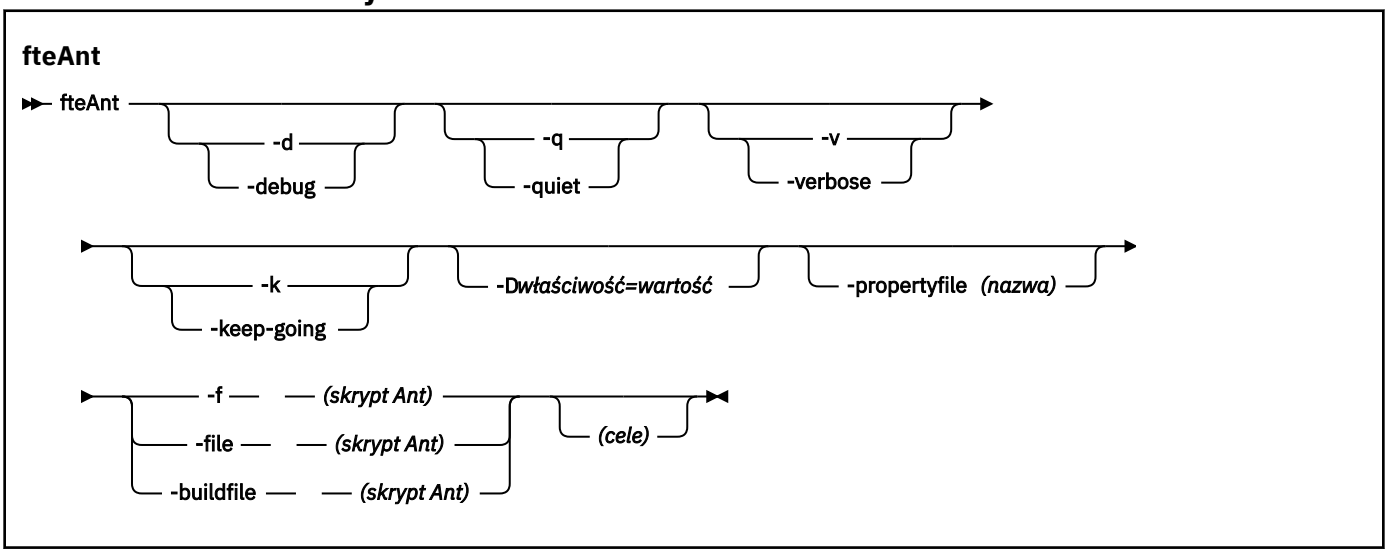
Komenda **`fteAnt`** uruchamia skrypty Ant w środowisku, w którym są dostępne zadania produktu Managed File Transfer Ant . W przeciwieństwie do standardowej komendy **`ant`** program **`fteAnt`** wymaga zdefiniowania pliku skryptu.

Zadania MFT Ant i parametry zagnieżdżone

Managed File Transfer udostępnia wiele zadań Ant , których można użyć w celu uzyskania dostępu do funkcji przesyłania plików. Dostępny jest również zestaw zagnieżdżonych parametrów. Parametry te opisują zagnieżdżone zestawy elementów, które są wspólne dla kilku dostarczonych zadań Ant.

W dalszej części tego tematu opisano składnię komendy **fteAnt** , parametry, przykład użycia i kody powrotu. Aby uzyskać szczegółowe informacje na temat zadań Ant i parametrów zagnieżdżonych, które są udostępniane przez MFT, Zapoznaj się z podtematami.

Składnia komendy fteAnt



Parametry

-debug lub -d
Opcjonalne. Generuj dane wyjściowe debugowania.

-quiet lub -q
Opcjonalne. Generuj minimalne dane wyjściowe.

-verbose lub -v
Opcjonalne. Generowanie szczegółowych danych wyjściowych.

-keep-go lub -k
Opcjonalne. Wykonaj wszystkie elementy docelowe, które nie zależą od elementów docelowych zakończonych niepowodzeniem.

-D *właściwość=wartość*
Opcjonalne. Należy użyć wartości *Wartość* dla danego parametru *Właściwość*. Właściwości ustawione za pomocą parametru **-D** mają pierwszeństwo przed właściwościami ustawionymi w pliku właściwości.

Użyj właściwości **com.ibm.wmqfte.propertyset** , aby określić zestaw opcji konfiguracyjnych, które są używane dla zadań Ant . Jako wartości tej właściwości należy użyć nazwy menedżera kolejek koordynacji innego niż domyślny. Następnie zadania Ant używają zestawu opcji konfiguracyjnych, które są powiązane z tym innym niż domyślny menedżerem kolejek koordynacji. Jeśli ta właściwość nie zostanie określona, zostanie użyty domyślny zestaw opcji konfiguracyjnych oparty na domyślnym menedżerze kolejek koordynacji. Jeśli atrybut **cmdqm** zostanie określony dla zadania Ant , ten atrybut ma pierwszeństwo przed zestawem opcji konfiguracyjnych określonym dla komendy **fteAnt** . To zachowanie ma zastosowanie niezależnie od tego, czy używany jest domyślny zestaw opcji konfiguracyjnych, czy też określono zestaw za pomocą właściwości **com.ibm.wmqfte.propertyset** .

-propertyfile (*nazwa*)
Opcjonalne. Załaduj wszystkie właściwości z pliku, przy czym właściwości **-D** mają pierwszeństwo.

-f (*skrypt Ant*), -file (*skrypt Ant*) lub -buildfile (*skrypt Ant*)
Wymagane. Określa nazwę skryptu Ant , który ma zostać uruchomiony.

cele
Opcjonalne. Nazwa jednego lub większej liczby celów do uruchomienia ze skryptu Ant. Jeśli wartość tego parametru nie zostanie podana, zostanie uruchomiony domyślny cel skryptu.

-version
Opcjonalne. Wyświetla komendę Managed File Transfer i wersje Ant .

-? lub -h
Opcjonalne. Wyświetla składnię komendy.

Przykład

W tym przykładzie uruchamiany jest cel **copy** w skrypcie Ant `fte_script.xml` , a komenda zapisuje dane wyjściowe debugowania na wyjściu standardowym.

```
fteAnt -d -f fte_script.xml copy
```

Kody powrotu

0
Wykonanie komendy zakończyło się pomyślnie.

1
Komenda została zakończona niepomyślnie.

Inne kody powrotu statusu można również określić w skryptach Ant, na przykład przy użyciu zadania Ant zakończonego niepowodzeniem.

Więcej informacji na ten temat zawiera sekcja Niepowodzenie .

Pojęcia pokrewne
Pierwsze kroki ze skryptami Ant w produkcie MFT
Zadania pokrewne
Używanie produktu Apache Ant z produktem MFT
Odsyłacze pokrewne
Przykładowe zadania programu Ant dla systemu MFT

fte: awaitoutcome Ant zadanie

Czeka na zakończenie operacji **fte:filecopy**, **fte:filemove**lub **fte:call** .

Atrybuty

Id
Wymagane. Identyfikuje operację przesyłania, która ma oczekiwać na wynik. Zwykle jest to właściwość ustawiona przez atrybut idProperty zadania fte: filecopy, fte: filemovelub fte: call .

rcproperty (właściwość rc)
Wymagane. Określa nazwę właściwości, w której ma zostać zapisany kod powrotu zadania **fte:awaitoutcome** .

limit czasu
Opcjonalne. Maksymalny czas (w sekundach) oczekiwania na zakończenie operacji. Minimalny limit czasu wynosi jedną sekundę. Jeśli wartość limitu czasu nie zostanie określona, zadanie **fte:awaitoutcome** będzie oczekiwać na określenie wyniku operacji w nieskończoność.

Przykład

W tym przykładzie zostanie uruchomiona kopia pliku, a jej identyfikator zostanie zapisany we właściwości `copy.id` . W trakcie kopiowania może być wykonywane inne przetwarzanie. Instrukcja **fte:awaitoutcome** jest używana do oczekiwania na zakończenie operacji kopiowania. Instrukcja **fte:awaitoutcome** określa, na którą operację należy czekać, używając identyfikatora zapisanego we

właściwości `copy.id` . **`fte:awaitoutcome`** zapisuje kod powrotu wskazujący wynik operacji kopiowania we właściwości o nazwie `copy.result`.

```
<-- issue a file copy request -->
<fte:filecopy
src="AGENT1@QM1"
dst="AGENT2@QM2"
idproperty="copy.id"
outcome="defer">

<fte:filespec
  srcfilespec="/home/fteuser1/file.bin"
  dstdir="/home/fteuser2"/>

</fte:filecopy>

<fte:awaitoutcome id="${copy.id}" rcProperty="copy.rc"/>

<echo>Copy id=${copy.id} rc=${copy.rc}</echo>
```

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

fte: wywołaj zadanie Ant

Zadania **`fte:call`** można użyć do zdalnego wywoływania skryptów i programów.

To zadanie umożliwia wysłanie żądania **`fte:call`** do agenta. Agent przetwarza to żądanie, uruchamiając skrypt lub program i zwracając wynik. Komendy do wywołania muszą być dostępne dla agenta. Upewnij się, że wartość właściwości commandPath w pliku `agent.properties` zawiera położenie komend do wywołania. Wszystkie informacje o ścieżce określone przez zagnieżdżony element komendy muszą być względne w stosunku do lokalizacji określonych przez właściwość commandPath . Domyślnie ścieżka commandPath jest pusta, dzięki czemu agent nie może wywoływać żadnych komend. Więcej informacji na temat tej właściwości zawiera sekcja commandPath MFT property.

Więcej informacji na temat pliku `agent.properties` zawiera sekcja Plik programu MFT `agent.properties`.

Atrybuty

agent
Wymagane. Określa agenta, do którego ma zostać wysłane żądanie **`fte:call`** . Podaj informacje o agencie w postaci: *agentname*`@`*qmgrname* , gdzie *agentname* jest nazwą agenta, a *qmgrname* jest nazwą menedżera kolejek, z którym ten agent jest bezpośrednio połączony.

cmdqm,
Opcjonalne. Menedżer kolejek komend, do którego ma zostać wysłane żądanie. Informacje te należy podać w postaci *qmgrname*`@`*host*`@`*port*`@`*channel*, gdzie:

- *qmgrname* jest nazwą menedżera kolejek
- *host* to opcjonalna nazwa hosta systemu, w którym działa menedżer kolejek.
- *port* to opcjonalny numer portu, na którym nasłuchuje menedżer kolejek.
- *channel* jest opcjonalnym kanałem SVRCONN, który ma być używany

Jeśli dla menedżera kolejek komend zostaną pominięte informacje *host*, *port*lub *channel* , zostaną użyte informacje o połączeniu określone w pliku `command.properties` .

Ostrzeżenie: Jeśli nie określono żadnej wartości dla:

- Zmienna *host* , używany jest tryb powiązań
- Zmienna *port* , używana jest wartość 1414
- *kanał* , zmienna SYSTEM.DEF.SVRCONN .

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties .

Nie można jednak pominąć atrybutów w środku, na przykład `qmgrname@host@@channel`. Może to być na przykład `qmgrname@host`, `qmgrname@host@port`lub `qmgrname@hostport@@channel`.

MFT dzieli dany atrybut przy użyciu separatora `@` . W zależności od liczby znalezionych znaczników pierwszy znacznik jest odbierany jako *qmgrname*, drugi jako *host*, trzeci jako *port* , a na końcu jako *kanał*.

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties.

Za pomocą właściwości **com.ibm.wmqfte.propertySet** można określić, który plik `command.properties` ma być używany. Więcej informacji na ten temat zawiera sekcja com.ibm.wmqfte.propertySet.

Jeśli atrybut cmdqm nie jest używany, zadanie domyślnie używa właściwości `com.ibm.wmqfte.ant.commandQueueManager` , jeśli ta właściwość jest ustawiona. Jeśli właściwość `com.ibm.wmqfte.ant.commandQueueManager` nie jest ustawiona, podejmowana jest próba nawiązania połączenia z domyślnym menedżerem kolejek zdefiniowanym w pliku `command.properties` . Format właściwości `com.ibm.wmqfte.ant.commandQueueManager` jest taki sam, jak atrybut cmdqm , czyli `qmgrname@host@port@channel`.

Właściwość idproperty
Parametr opcjonalny, chyba że określono parametr `outcome` o wartości `defer`. Określa nazwę właściwości, do której ma zostać przypisany identyfikator przesyłania. Identyfikatory przesyłania są generowane w momencie przesłania żądania przesyłania i można ich użyć do śledzenia postępu przesyłania, diagnozowania problemów z przesyłaniem i anulowania przesyłania.

Nie można określić tej właściwości, jeśli określono również właściwość `outcome` o wartości `ignore`. Należy jednak podać wartość `idproperty` , jeśli określono również właściwość `outcome` o wartości `defer`.

JOBNAME
Opcjonalne. Przypisuje nazwę zadania do żądania **fte:call** . Za pomocą nazw zadań można tworzyć logiczne grupy transferów. Zadanie “Zadanie fte: uuid Ant” na stronie 2180 służy do generowania pseuunikalnych nazw zadań. Jeśli atrybut jobname nie jest używany, zadanie domyślnie używa wartości właściwości `com.ibm.wmqfte.ant.jobName` , jeśli ta właściwość jest ustawiona. Jeśli ta właściwość nie zostanie ustawiona, z żądaniem **fte:call** nie będzie powiązana żadna nazwa zadania.

Origuser
Opcjonalne. Określa identyfikator użytkownika inicjującego, który ma zostać powiązany z żądaniem **fte:call** . Jeśli atrybut origuser nie jest używany, zadanie domyślnie używa identyfikatora użytkownika, który jest używany do uruchamiania skryptu Ant.

wynik
Opcjonalne. Określa, czy zadanie oczekuje na zakończenie operacji **fte:call** przed zwróceniem sterowania do skryptu Ant . Podaj jedną z następujących opcji:

Oczekiwanie
Zadanie oczekuje na zakończenie operacji **fte:call** przed zwróceniem. Jeśli zostanie podany atrybut `outcome` o wartości `await` , atrybut `idproperty` jest opcjonalny.

defer
Zadanie jest zwracane natychmiast po wysłaniu żądania **fte:call** i zakłada, że wynik operacji wywołania zostanie później rozpatrzony przy użyciu zadania awaitoutcome lub ignoreoutcome . Jeśli podano atrybut `outcome` o wartości `defer` , wymagany jest atrybut `idproperty` .

ignoruj
Jeśli wynik operacji **fte:call** nie jest istotny, można podać wartość `ignore`. Zadanie jest następnie zwracane natychmiast po wprowadzeniu żądania **fte:call** bez przydzielania zasobów na potrzeby śledzenia wyniku komendy. Jeśli podano atrybut `outcome` o wartości `ignore` , nie można podać atrybutu `idproperty` .

Jeśli atrybut outcome nie zostanie określony, domyślnie przyjmowana jest wartość `await`.

Właściwość rc
Opcjonalne. Określa nazwę właściwości, do której ma zostać przypisany kod wyniku żądania **fte:call** . Kod wyniku odzwierciedla ogólny wynik żądania **fte:call** .

Nie można określić tej właściwości, jeśli określono również właściwość `outcome` o wartości `ignore` lub `defer`. Jeśli jednak określono wynik `await`, należy podać wartość `rcproperty` .

Parametry określone jako elementy zagnieżdżone

fte: komenda
Określa komendę wywołaną przez agenta. Z daną operacją**fte:call** można powiązać tylko jeden element `fte:command` . Komenda do wywołania musi znajdować się w ścieżce określonej przez właściwość commandPath w pliku `agent.properties` agenta.

fte: metadane
Istnieje możliwość określenia metadanych, które mają zostać powiązane z operacją wywołania. Te metadane są zapisywane w komunikatach dziennika generowanych przez operację wywołania. Z danym elementem przesyłania można powiązać tylko jeden blok metadanych. Ten blok może jednak zawierać wiele elementów metadanych.

Przykład

W tym przykładzie przedstawiono sposób wywołania komendy w AGENT1 działającej w menedżerze kolejek QM1. Komenda do wywołania to skrypt `command.sh`, a skrypt jest wywoływany z pojedynczym argumentem `xyz`. Komenda `command.sh` znajduje się w ścieżce określonej przez właściwość commandPath w pliku `agent.properties` agenta.

```
<fte:call cmdqm="QM0@localhost@1414@SYSTEM.DEF.SVRCONN"
            agent="AGENT1@QM1"
            rcproperty="call.rc"
            origuser="bob"
            jobname="${job.id}">

    <fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30">
        <fte:arg value="xyz"/>
    </fte:command>

    <fte:metadata>
        <fte:entry name="org.foo.accountName" value="BDG3R"/>
    </fte:metadata>

</fte:call>
```

Zadania pokrewne
Używanie produktu Apache Ant z produktem MFT

fte: anuluj zadanie Ant

Anuluje przesyłanie zarządzane przez Managed File Transfer lub wywołanie zarządzane. Przesyłanie zarządzane mogło zostać utworzone przy użyciu zadań **fte:filecopy** lub **fte:filemove** . Wywołanie zarządzane mogło zostać utworzone przy użyciu zadania **fte:call** .

Atrybuty

agent
Wymagane. Określa agenta, do którego ma zostać wysłane żądanie **fte:cancel** . Wartość ma postać: *agentname*@*qmgrname* , gdzie *agentname* jest nazwą agenta, a *qmgrname* jest nazwą menedżera kolejek, z którym ten agent jest bezpośrednio połączony.

cmdqm,
Opcjonalne. Menedżer kolejek komend, do którego ma zostać wysłane żądanie. Informacje te należy podać w postaci *qmgrname*@*host*@*port*@*channel*, gdzie:

- *qmgrname* jest nazwą menedżera kolejek

- *host* to opcjonalna nazwa hosta systemu, w którym działa menedżer kolejek.
- *port* to opcjonalny numer portu, na którym nasłuchuje menedżer kolejek.
- *channel* jest opcjonalnym kanałem SVRCONN, który ma być używany

Jeśli dla menedżera kolejek komend zostaną pominięte informacje *host*, *port*lub *channel* , zostaną użyte informacje o połączeniu określone w pliku command.properties .

Ostrzeżenie: Jeśli nie określono żadnej wartości dla:

- Zmienna *host* , używany jest tryb powiązań
- Zmienna *port* , używana jest wartość 1414
- *kanał* , zmienna SYSTEM.DEF.SVRCONN .

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties .

Nie można jednak pominąć atrybutów w środku, na przykład qmgrname@host@@channel. Może to być na przykład qmgrname@host, qmgrname@host@portlub qmgrname@hostport@@channel.

MFT dzieli dany atrybut przy użyciu separatora @ . W zależności od liczby znalezionych znaczników pierwszy znacznik jest odbierany jako *qmgrname*, drugi jako *host*, trzeci jako *port* , a na końcu jako *kanał*.

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties.

Za pomocą właściwości **com.ibm.wmqfte.propertySet** można określić, który plik command.properties ma być używany. Więcej informacji na ten temat zawiera sekcja com.ibm.wmqfte.propertySet.

Jeśli atrybut cmdqm nie jest używany, zadanie domyślnie używa właściwości com.ibm.wmqfte.ant.commandQueueManager , jeśli ta właściwość jest ustawiona. Jeśli właściwość com.ibm.wmqfte.ant.commandQueueManager nie jest ustawiona, podejmowana jest próba nawiązania połączenia z domyślnym menedżerem kolejek zdefiniowanym w pliku command.properties . Format właściwości com.ibm.wmqfte.ant.commandQueueManager jest taki sam, jak atrybut cmdqm , czyli qmgrname@host@port@channel.

Id

Wymagane. Określa identyfikator przesyłania, które ma zostać anulowane. Identyfikatory przesyłania są generowane w punkcie, w którym żądanie przesyłania jest przesyłane przez zadania fte: filecopy i fte: filemove .

origuser (użytkownik origuser)

Opcjonalne. Określa identyfikator użytkownika inicjującego, który ma zostać powiązany z żądaniem **cancel** . Jeśli atrybut origuser nie jest używany, domyślnie zadanie używa identyfikatora użytkownika, który jest używany do uruchamiania skryptu Ant .

Przykład

Przykład wysyła żądanie **fte:cancel** do menedżera kolejek komend qm0. Celem żądania **fte:cancel** jest agent1 w menedżerze kolejek qm1 dla identyfikatora przesyłania zapełnionego przez zmienną transfer.id . Żądanie jest uruchamiane przy użyciu identyfikatora użytkownika "bob" .

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
            agent="agent1@qm1"
            id="${transfer.id}"
            origuser="bob"/>
```

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

fte: filecopy Ant zadanie

Zadanie **fte:filecopy** kopiuje pliki między agentami Managed File Transfer . Plik nie został usunięty z agenta źródłowego.

Atrybuty

cmdqm,

Opcjonalne. Menedżer kolejek komend, do którego ma zostać wysłane żądanie. Informacje te należy podać w postaci *qmgrname*@*host*@*port*@*channel*, gdzie:

- *qmgrname* jest nazwą menedżera kolejek
- *host* to opcjonalna nazwa hosta systemu, w którym działa menedżer kolejek.
- *port* to opcjonalny numer portu, na którym nasłuchuje menedżer kolejek.
- *channel* jest opcjonalnym kanałem SVRCONN, który ma być używany

Jeśli dla menedżera kolejek komend zostaną pominięte informacje *host*, *port*lub *channel* , zostaną użyte informacje o połączeniu określone w pliku command.properties .

Ostrzeżenie: Jeśli nie określono żadnej wartości dla:

- Zmienna *host* , używany jest tryb powiązań
- Zmienna *port* , używana jest wartość 1414
- *kanał* , zmienna SYSTEM.DEF.SVRCONN .

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties .

Nie można jednak pominąć atrybutów w środku, na przykład qmgrname@host@@channel. Może to być na przykład qmgrname@host, qmgrname@host@portlub qmgrname@hostport@@channel.

MFT dzieli dany atrybut przy użyciu separatora @ . W zależności od liczby znalezionych znaczników pierwszy znacznik jest odbierany jako *qmgrname*, drugi jako *host*, trzeci jako *port* , a na końcu jako *kanał*.

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties.

Za pomocą właściwości **com.ibm.wmqfte.propertySet** można określić, który plik command.properties ma być używany. Więcej informacji na ten temat zawiera sekcja com.ibm.wmqfte.propertySet.

Jeśli atrybut cmdqm nie jest używany, zadanie domyślnie używa właściwości com.ibm.wmqfte.ant.commandQueueManager , jeśli ta właściwość jest ustawiona. Jeśli właściwość com.ibm.wmqfte.ant.commandQueueManager nie jest ustawiona, podejmowana jest próba nawiązania połączenia z domyślnym menedżerem kolejek zdefiniowanym w pliku command.properties . Format właściwości com.ibm.wmqfte.ant.commandQueueManager jest taki sam, jak atrybut cmdqm , czyli qmgrname@host@port@channel.

Dst

Wymagane. Określa agenta docelowego dla operacji kopiowania. Podaj następujące informacje w postaci: agentname@qmgrname , gdzie agentname jest nazwą agenta docelowego, a qmgrname jest nazwą menedżera kolejek, z którym ten agent jest bezpośrednio połączony.

Właściwość idproperty

Parametr opcjonalny, chyba że określono parametr outcome o wartości defer. Określa nazwę właściwości, do której ma zostać przypisany identyfikator przesyłania. Identyfikatory przesyłania są generowane w momencie przesłania żądania przesyłania i można ich użyć do śledzenia postępu przesyłania, diagnozowania problemów z przesyłaniem i anulowania przesyłania.

Nie można określić tej właściwości, jeśli określono również właściwość outcome o wartości ignore. Należy jednak podać wartość idproperty , jeśli określono również właściwość outcome o wartości defer.

JOBNAME
Opcjonalne. Przypisuje nazwę zadania do żądania kopiowania. Za pomocą nazw zadań można tworzyć logiczne grupy transferów. Zadanie "Zadanie fte: uuid Ant" na stronie 2180 służy do generowania pseuunikalnych nazw zadań. Jeśli atrybut jobname nie jest używany, zadanie domyślnie używa wartości właściwości `com.ibm.wmqfte.ant.jobName` , jeśli ta właściwość jest ustawiona. Jeśli ta właściwość nie zostanie ustawiona, z żądaniem kopiowania nie będzie powiązana żadna nazwa zadania.

Origuser
Opcjonalne. Określa identyfikator użytkownika inicjującego, który ma zostać powiązany z żądaniem kopiowania. Jeśli atrybut origuser nie jest używany, zadanie domyślnie używa identyfikatora użytkownika, który jest używany do uruchamiania skryptu Ant .

wynik
Opcjonalne. Określa, czy zadanie oczekuje na zakończenie operacji kopiowania przed zwróceniem kontroli do skryptu Ant . Podaj jedną z następujących opcji:

Oczekiwanie
Zadanie oczekuje na zakończenie operacji kopiowania przed powrotem. Jeśli zostanie podany atrybut `outcome` o wartości `await` , atrybut `idproperty` jest opcjonalny.

defer
Zadanie jest zwracane natychmiast po przesłaniu żądania kopiowania i przyjmuje, że wynik operacji kopiowania jest traktowany w późniejszym czasie przy użyciu zadania "fte: awaitoutcome Ant zadanie" na stronie 2166 lub "fte: ignoreoutcome Ant zadanie" na stronie 2178 . Jeśli podano atrybut `outcome` o wartości `defer` , wymagany jest atrybut `idproperty` .

ignoruj
Jeśli wynik operacji kopiowania nie jest istotny, można podać wartość `ignore`. Następnie zadanie powraca natychmiast po wysłaniu żądania kopiowania bez przydzielania zasobów na potrzeby śledzenia wyniku przesyłania. Jeśli podano atrybut `outcome` o wartości `ignore` , nie można podać atrybutu `idproperty` .

Jeśli atrybut outcome nie zostanie określony, domyślnie przyjmowana jest wartość `await`.

priorytet
Opcjonalne. Określa priorytet, który ma być powiązany z żądaniem kopiowania. Ogólnie rzecz biorąc, żądania przesyłania o wyższym priorytecie mają pierwszeństwo przed żądaniami o niższym priorytecie. Wartość priorytetu musi być z zakresu od 0 do 9 (włącznie). Wartość priorytetu 0 jest najniższym priorytetem, a wartość 9 jest najwyższym priorytetem. Jeśli atrybut `priority` nie zostanie określony, wartością domyślną przesyłania będzie priorytet 0.

Właściwość rc
Opcjonalne. Określa nazwę właściwości, do której ma zostać przypisany kod wyniku żądania kopiowania. Kod wyniku odzwierciedla ogólny wynik żądania kopiowania.

Nie można określić tej właściwości, jeśli określono również właściwość `outcome` o wartości `ignore` lub `defer`. Jeśli jednak zostanie określony wynik `await`, należy podać wartość `rcproperty` .

Limit czasu transferRecovery
Opcjonalne. Ustawia czas (w sekundach), podczas którego agent źródłowy próbuje odzyskać wstrzymane przesyłanie plików. Podaj jedną z następujących opcji:

-1
Agent będzie próbował odzyskać wstrzymane przesyłanie do momentu zakończenia przesyłania. Użycie tej opcji jest równoważne domyślnemu zachowaniu agenta, gdy właściwość nie jest ustawiona.

0
Agent zatrzymuje przesyłanie plików natychmiast po zakończeniu odtwarzania.

>0
Agent kontynuuje próbę odzyskania wstrzymanego przesyłania przez czas w sekundach, ustawiony przez podaną dodatnią liczbę całkowitą. Na przykład składnia

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
```

```
                rcproperty="copy.result" transferRecoveryTimeout="21600">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filecopy>
```

Wskazuje, że agent będzie próbował odzyskać przesyłanie przez 6 godzin od momentu, gdy zostanie uruchomiony proces odtwarzania. Maksymalna wartość tego atrybutu to 999999999.

Określenie w ten sposób wartości limitu czasu odtwarzania przesyłania powoduje, że jest ona ustawiana dla każdej operacji przesyłania. Aby ustawić wartość globalną dla wszystkich transferów w sieci Managed File Transfer , można dodać właściwość do pola Właściwości limitu czasu odtwarzania przesyłania. Więcej informacji na ten temat zawiera sekcja Opcja limitu czasu dla operacji przesyłania podczas odtwarzania.

src
Wymagane. Określa agenta źródłowego dla operacji kopiowania. Podaj następujące informacje w postaci: *nazwa_agenta*@*nazwa_menedżera_kolejek* , gdzie *nazwa_agenta* jest nazwą agenta źródłowego, a *nazwa_menedżera_kolejek* jest nazwą menedżera kolejek, z którym ten agent jest bezpośrednio połączony.

Parametry określone jako elementy zagnieżdżone

fte: specyfikacja_pliku
Wymagane. Należy określić co najmniej jedną specyfikację pliku, która identyfikuje pliki do skopiowania. W razie potrzeby można określić więcej niż jedną specyfikację pliku. Więcej informacji można znaleźć w sekcji “fte: filespec Ant element zagnieżdżony” na stronie 2180.

fte: metadane
Istnieje możliwość określenia metadanych, które mają zostać powiązane z operacją kopiowania. Te metadane są przenoszone wraz z przesyłaniem i są rejestrowane w komunikatach dziennika generowanych przez przesyłanie. Z danym elementem przesyłania można powiązać tylko jeden blok metadanych. Ten blok może jednak zawierać wiele elementów metadanych. Więcej informacji na ten temat zawiera sekcja fte: metadata .

fte: presrc,
Określa wywołanie programu w agencie źródłowym przed rozpoczęciem przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:presrc` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

fte: predst,
Określa wywołanie programu, które ma zostać przeprowadzone w agencie docelowym przed rozpoczęciem przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:predst` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

fte: postsrc,
Określa wywołanie programu w agencie źródłowym po zakończeniu przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:postsrc` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

fte: postdst
Określa wywołanie programu, które ma zostać wykonane w agencie docelowym po zakończeniu przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:postdst` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

Jeśli komendy fte:presrc, fte:predst, fte:postsrc, fte:postdst i wyjścia nie zwracają statusu powodzenia, reguły są następujące w podanej kolejności:

1. Uruchom źródłowe wyjścia początkowe. Jeśli wyjścia uruchamiania źródła nie powiodą się, przesyłanie nie powiedzie się i nie zostaną uruchomione żadne dalsze działania.
2. Uruchom wywołanie przed kodem źródłowym (jeśli istnieje). Jeśli wywołanie przed kodem źródłowym nie powiedzie się, przesyłanie nie powiedzie się i nie będzie wykonywane żadne dalsze działanie.

3. Uruchom docelowe wyjścia początkowe. Jeśli uruchomienie wyjścia docelowego nie powiedzie się, przesyłanie nie powiedzie się i nie będzie wykonywane żadne dalsze działanie.
4. Uruchom wywołanie przed miejscem docelowym (jeśli istnieje). Jeśli wywołanie przed miejscem docelowym nie powiedzie się, przesyłanie nie powiedzie się i nie będzie wykonywane żadne dalsze działanie.
5. Wykonaj przesyłanie plików.
6. Uruchom docelowe wyjścia końcowe. Dla tych wyjść nie ma statusu niepowodzenia.
7. Jeśli przesyłanie powiedzie się (jeśli przesyłanie niektórych plików powiedzie się, zostanie ono uznane za pomyślne), należy uruchomić wywołanie po przesłaniu (jeśli istnieje). Jeśli wywołanie po miejscu docelowym nie powiedzie się, przesyłanie nie powiedzie się.
8. Uruchom źródłowe wyjścia końcowe. Dla tych wyjść nie ma statusu niepowodzenia.
9. Jeśli przesyłanie zakończyło się pomyślnie, uruchom wywołanie po źródle (jeśli istnieje). Jeśli wywołanie po źródle nie powiedzie się, przesyłanie nie powiedzie się.

Przykłady

W tym przykładzie przedstawiono podstawowe przesyłanie plików między systemami `agent1` i `agent2`. Komenda uruchamiającą przesyłanie plików jest wysyłana do menedżera kolejek o nazwie `qm0`, przy użyciu połączenia w trybie transportu klienta. Wynik operacji przesyłania pliku jest przypisywany do właściwości o nazwie `copy.result`.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="copy.result">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filecopy>
```

W tym przykładzie przedstawiono tę samą operację przesyłania plików, ale z dodaniem metadanych i uruchomieniem programu w agencie źródłowym po zakończeniu przesyłania.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm"1 dst="agent2@qm2"
              rcproperty="copy.result">


    <fte:metadata>
            <fte:entry name="org.example.departId" value="ACCOUNTS"/>
            <fte:entry name="org.example.batchGroup" value="A1"/>
    </fte:metadata>

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

    <fte:postsrc command="/home/fteuser2/scripts/post.sh" successrc="1" >
            <fte:arg value="/home/fteuser2/file.bin"/>
  </fte:postsrc>
</fte:filecopy>
```

Pojęcia pokrewne

Opcja limitu czasu dla przesyłania plików podczas odtwarzania

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

fte: filemove Ant zadanie

Zadanie **`fte:filemove`** przenosi pliki między agentami Managed File Transfer . Po pomyślnym przesłaniu pliku z agenta źródłowego do agenta docelowego plik jest usuwany z agenta źródłowego.

Atrybuty

cmdqm,

Opcjonalne. Menedżer kolejek komend, do którego ma zostać wysłane żądanie. Informacje te należy podać w postaci *qmgrname*@*host*@*port*@*channel*, gdzie:

- *qmgrname* jest nazwą menedżera kolejek
- *host* to opcjonalna nazwa hosta systemu, w którym działa menedżer kolejek.
- *port* to opcjonalny numer portu, na którym nasłuchuje menedżer kolejek.
- *channel* jest opcjonalnym kanałem SVRCONN, który ma być używany

Jeśli dla menedżera kolejek komend zostaną pominięte informacje *host*, *port*lub *channel* , zostaną użyte informacje o połączeniu określone w pliku `command.properties` .

Ostrzeżenie: Jeśli nie określono żadnej wartości dla:

- Zmienna *host* , używany jest tryb powiązań
- Zmienna *port* , używana jest wartość 1414
- *kanał* , zmienna SYSTEM.DEF.SVRCONN .

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties .

Nie można jednak pominąć atrybutów w środku, na przykład `qmgrname@host@@channel`. Może to być na przykład `qmgrname@host`, `qmgrname@host@portlub qmgrname@hostport@@channel`.

MFT dzieli dany atrybut przy użyciu separatora `@` . W zależności od liczby znalezionych znaczników pierwszy znacznik jest odbierany jako *qmgrname*, drugi jako *host*, trzeci jako *port* , a na końcu jako *kanał*.

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties.

Za pomocą właściwości **`com.ibm.wmqfte.propertySet`** można określić, który plik `command.properties` ma być używany. Więcej informacji na ten temat zawiera sekcja com.ibm.wmqfte.propertySet.

Jeśli atrybut cmdqm nie jest używany, zadanie domyślnie używa właściwości `com.ibm.wmqfte.ant.commandQueueManager` , jeśli ta właściwość jest ustawiona. Jeśli właściwość `com.ibm.wmqfte.ant.commandQueueManager` nie jest ustawiona, podejmowana jest próba nawiązania połączenia z domyślnym menedżerem kolejek zdefiniowanym w pliku `command.properties` . Format właściwości `com.ibm.wmqfte.ant.commandQueueManager` jest taki sam, jak atrybut cmdqm , czyli `qmgrname@host@port@channel`.

Dst

Wymagane. Określa agenta docelowego dla operacji kopiowania. Podaj następujące informacje w postaci: *agentname*@*qmgrname* , gdzie *agentname* jest nazwą agenta docelowego, a *qmgrname* jest nazwą menedżera kolejek, z którym ten agent jest bezpośrednio połączony.

Właściwość idproperty

Parametr opcjonalny, chyba że określono parametr `outcome` o wartości `defer`. Określa nazwę właściwości, do której ma zostać przypisany identyfikator przesyłania. Identyfikatory przesyłania są generowane w momencie przesłania żądania przesyłania i można ich użyć do śledzenia postępu przesyłania, diagnozowania problemów z przesyłaniem i anulowania przesyłania.

Nie można określić tej właściwości, jeśli określono również właściwość `outcome` o wartości `ignore`. Należy jednak podać wartość `idproperty` , jeśli określono również właściwość `outcome` o wartości `defer`.

JOBNAME

Opcjonalne. Przypisuje nazwę zadania do żądania przeniesienia. Za pomocą nazw zadań można tworzyć logiczne grupy transferów. Użyj zadania fte: uuid , aby wygenerować pseudounikalne nazwy zadań. Jeśli atrybut jobname nie jest używany, zadanie domyślnie używa wartości właściwości `com.ibm.wmqfte.ant.jobName` , jeśli ta właściwość jest ustawiona. Jeśli ta właściwość nie zostanie ustawiona, z żądaniem przeniesienia nie będzie powiązana żadna nazwa zadania.

Origuser
Opcjonalne. Określa identyfikator użytkownika inicjującego, który ma zostać powiązany z żądaniem przeniesienia. Jeśli atrybut origuser nie jest używany, zadanie domyślnie używa identyfikatora użytkownika, który jest używany do uruchamiania skryptu Ant .

wynik
Opcjonalne. Określa, czy zadanie oczekuje na zakończenie operacji przenoszenia przed zwróceniem kontroli do skryptu Ant . Podaj jedną z następujących opcji:

Oczekiwanie
Zadanie oczekuje na zakończenie operacji przenoszenia przed powrotem. Jeśli zostanie podany atrybut `outcome` o wartości `await` , atrybut `idproperty` jest opcjonalny.

defer
Zadanie powraca natychmiast po wprowadzeniu żądania przeniesienia i przyjmuje, że wynik operacji przeniesienia jest traktowany w późniejszym czasie przy użyciu zadania “fte: awaitoutcome Ant zadanie” na stronie 2166 lub “fte: ignoreoutcome Ant zadanie” na stronie 2178 . Jeśli podano atrybut `outcome` o wartości `defer` , wymagany jest atrybut `idproperty` .

ignoruj
Jeśli wynik operacji przenoszenia nie jest istotny, można określić wartość `ignore`. Następnie zadanie powraca natychmiast po wprowadzeniu żądania przeniesienia, bez przydzielania zasobów na potrzeby śledzenia wyniku przesyłania. Jeśli podano atrybut `outcome` o wartości `ignore` , nie można podać atrybutu `idproperty` .

Jeśli atrybut outcome nie zostanie określony, domyślnie przyjmowana jest wartość `await`.

priorytet
Opcjonalne. Określa priorytet, który ma zostać powiązany z żądaniem przeniesienia. Ogólnie rzecz biorąc, żądania przesyłania o wyższym priorytecie mają pierwszeństwo przed żądaniami o niższym priorytecie. Wartość priorytetu musi być z zakresu od 0 do 9 (włącznie). Wartość priorytetu 0 jest najniższym priorytetem, a wartość 9 jest najwyższym priorytetem. Jeśli atrybut `priority` nie zostanie określony, wartością domyślną przesyłania będzie priorytet 0.

Właściwość rc
Opcjonalne. Określa nazwę właściwości, do której ma zostać przypisany kod wyniku żądania przeniesienia. Kod wyniku odzwierciedla ogólny wynik zlecenia przeniesienia.

Nie można określić tej właściwości, jeśli określono również właściwość `outcome` o wartości `ignore` lub `defer`. Jeśli jednak określono wynik `await`, należy podać wartość `rcproperty` .

Limit czasu transferRecovery
Opcjonalne. Ustawia czas (w sekundach), podczas którego agent źródłowy próbuje odzyskać wstrzymane przesyłanie plików. Podaj jedną z następujących opcji:

-1
Agent będzie próbował odzyskać wstrzymane przesyłanie do momentu zakończenia przesyłania. Użycie tej opcji jest równoważne domyślnemu zachowaniu agenta, gdy właściwość nie jest ustawiona.

0
Agent zatrzymuje przesyłanie plików natychmiast po zakończeniu odtwarzania.

>0
Agent kontynuuje próbę odzyskania wstrzymanego przesyłania przez czas w sekundach, ustawiony przez podaną dodatnią liczbę całkowitą. Na przykład składnia

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src=agent1@qm1 dst="agent2@qm2"
              rcproperty="move.result" transferRecoveryTimeout="21600">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/
file.bin"/>

</fte:filemove
```

Wskazuje, że agent będzie próbował odzyskać przesyłanie przez 6 godzin od momentu, gdy zostanie uruchomiony proces odtwarzania. Maksymalna wartość tego atrybutu to 999999999.

Określenie w ten sposób wartości limitu czasu odtwarzania przesyłania powoduje, że jest ona ustawiana dla każdej operacji przesyłania. Aby ustawić wartość globalną dla wszystkich transferów w sieci Managed File Transfer , można dodać właściwość do pola Właściwości limitu czasu odtwarzania przesyłania. Więcej informacji na ten temat zawiera sekcja Opcja limitu czasu dla operacji przesyłania podczas odtwarzania.

src
Wymagane. Określa agent źródłowy dla operacji przenoszenia. Podaj następujące informacje w postaci: *agentname*@*qmgrname* , gdzie *agentname* jest nazwą agenta źródłowego, a *qmgrname* jest nazwą menedżera kolejek, z którym ten agent jest połączony bezpośrednio.

Parametry określone jako elementy zagnieżdżone

fte: specyfikacja_pliku
Wymagane. Należy określić co najmniej jedną specyfikację pliku, która identyfikuje pliki do przeniesienia. W razie potrzeby można określić więcej niż jedną specyfikację pliku. Więcej informacji można znaleźć w sekcji “fte: filespec Ant element zagnieżdżony” na stronie 2180.

fte: metadane
Opcjonalne. Istnieje możliwość określenia metadanych, które mają zostać powiązane z operacją przenoszenia pliku. Te metadane są przenoszone wraz z przesyłaniem i są rejestrowane w komunikatach dziennika generowanych przez przesyłanie. Z danym elementem przesyłania można powiązać tylko jeden blok metadanych. Ten blok może jednak zawierać wiele elementów metadanych. Więcej informacji na ten temat zawiera sekcja fte: metadata .

fte: presrc,
Opcjonalne. Określa wywołanie programu w agencie źródłowym przed rozpoczęciem przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:presrc` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

fte: predst,
Opcjonalne. Określa wywołanie programu, które ma zostać przeprowadzone w agencie docelowym przed rozpoczęciem przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:predst` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

fte: postsrc,
Opcjonalne. Określa wywołanie programu w agencie źródłowym po zakończeniu przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:postsrc` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

fte: postdst
Opcjonalne. Określa wywołanie programu, które ma zostać wykonane w agencie docelowym po zakończeniu przesyłania. Z danym przesyłaniem można powiązać tylko jeden element `fte:postdst` . Więcej informacji na ten temat zawiera sekcja Wywołanie programu .

Jeśli komendy fte:presrc, fte:predst, fte:postsrc, fte:postdst i wyjścia nie zwracają statusu powodzenia, reguły są następujące w podanej kolejności:

1. Uruchom źródłowe wyjścia początkowe. Jeśli wyjścia uruchamiania źródła nie powiodą się, przesyłanie nie powiedzie się i nie zostaną uruchomione żadne dalsze działania.
2. Uruchom wywołanie przed kodem źródłowym (jeśli istnieje). Jeśli wywołanie przed kodem źródłowym nie powiedzie się, przesyłanie nie powiedzie się i nie będzie wykonywane żadne dalsze działanie.
3. Uruchom docelowe wyjścia początkowe. Jeśli uruchomienie wyjścia docelowego nie powiedzie się, przesyłanie nie powiedzie się i nie będzie wykonywane żadne dalsze działanie.
4. Uruchom wywołanie przed miejscem docelowym (jeśli istnieje). Jeśli wywołanie przed miejscem docelowym nie powiedzie się, przesyłanie nie powiedzie się i nie będzie wykonywane żadne dalsze działanie.
5. Wykonaj przesyłanie plików.
6. Uruchom docelowe wyjścia końcowe. Dla tych wyjść nie ma statusu niepowodzenia.

7. Jeśli przesyłanie powiedzie się (jeśli niektóre pliki zostały pomyślnie przesłane, przesyłanie zostanie uznane za pomyślne), należy uruchomić wywołanie po przesłaniu do miejsca docelowego (jeśli istnieje). Jeśli wywołanie po miejscu docelowym nie powiedzie się, przesyłanie nie powiedzie się.
8. Uruchom źródłowe wyjścia końcowe. Dla tych wyjść nie ma statusu niepowodzenia.
9. Jeśli przesyłanie powiodło się, uruchom wywołanie po źródle (jeśli istnieje). Jeśli wywołanie po źródle nie powiedzie się, przesyłanie nie powiedzie się.

Przykłady

W tym przykładzie przedstawiono podstawowe przenoszenie plików między systemami `agent1` i `agent2`. Komenda uruchamiającą przenoszenie plików jest wysyłana do menedżera kolejek o nazwie `qm0`, przy użyciu połączenia w trybie transportu klienta. Wynik operacji przesyłania pliku jest przypisywany do właściwości o nazwie `move.result`.

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="move.result">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filemove>
```

Pojęcia pokrewne

Opcja limitu czasu dla przesyłania plików podczas odtwarzania

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

fte: ignoreoutcome Ant zadanie

Zignoruj wynik komendy **fte:filecopy**, **fte:filemove**lub **fte:call** . Po określeniu zadania **fte:filecopy**, **fte:filemove**lub **fte:call** , które ma mieć wynik `defer`, zadanie Ant przydziela zasoby do śledzenia tego wyniku. Jeśli użytkownik nie jest już zainteresowany wynikiem, może użyć zadania **fte:ignoreoutcome** , aby zwolnić te zasoby.

Atrybuty

Id

Wymagane. Identyfikuje wynik, który nie jest już przedmiotem zainteresowania. Zwykle identyfikator ten określa się za pomocą właściwości ustawionej za pomocą atrybutu `idproperty` zadania “fte: filecopy Ant zadanie” na stronie 2171, “fte: filemove Ant zadanie” na stronie 2174lub “fte: wywołaj zadanie Ant” na stronie 2167 .

Przykład

W tym przykładzie przedstawiono sposób użycia zadania fte: ignoreoutcome w celu zwolnienia zasobów przydzielonych do śledzenia wyniku wcześniejszego zadania “fte: filecopy Ant zadanie” na stronie 2171 .

```
<!-- issue a file copy request -->
<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
                  src="agent1@qm1" dst="agent1@qm1"
                  idproperty="copy.id"
                  outcome="defer"/>

<!-- do some other things -->

<!-- decide that the result of the copy is not interesting -->
<fte:ignoreoutcome id="${copy.id}"/>
```

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

fte: zadanie ping Ant

To zadanie programu IBM MQ Managed File Transfer Ant przesyła do agenta komendę ping w celu uzyskania odpowiedzi i określa, czy agent może przetwarzać operacje przesyłania.

Uwaga: IBM WebSphere MQ File Transfer Edition (FTE) nie jest już obsługiwanym produktem. Informacje o migrowaniu FTE do składnika Managed File Transfer w programie IBM MQ można znaleźć w sekcji Migrowanie produktu Managed File Transfer.

Atrybuty

agent
Wymagane. Określa agenta, do którego ma zostać wysłane żądanie **fte:ping** . Wartość ma postać: *agentname*@*qmgrname* , gdzie *agentname* jest nazwą agenta, a *qmgrname* jest nazwą menedżera kolejek, z którym ten agent jest bezpośrednio połączony.

cmdqm,
Opcjonalne. Menedżer kolejek komend, do którego ma zostać wysłane żądanie. Informacje te należy podać w postaci *qmgrname*@*host*@*port*@*channel*, gdzie:

- *qmgrname* jest nazwą menedżera kolejek
- *host* to opcjonalna nazwa hosta systemu, w którym działa menedżer kolejek.
- *port* to opcjonalny numer portu, na którym nasłuchuje menedżer kolejek.
- *channel* jest opcjonalnym kanałem SVRCONN, który ma być używany

Jeśli dla menedżera kolejek komend zostaną pominięte informacje *host*, *port*lub *channel* , zostaną użyte informacje o połączeniu określone w pliku command.properties .

Ostrzeżenie: Jeśli nie określono żadnej wartości dla:

- Zmienna *host* , używany jest tryb powiązań
- Zmienna *port* , używana jest wartość 1414
- *kanał* , zmienna SYSTEM.DEF.SVRCONN .

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties .

Nie można jednak pominąć atrybutów w środku, na przykład qmgrname@host@@channel. Może to być na przykład qmgrname@host, qmgrname@host@portlub qmgrname@hostport@@channel.

MFT dzieli dany atrybut przy użyciu separatora @ . W zależności od liczby znalezionych znaczników pierwszy znacznik jest odbierany jako *qmgrname*, drugi jako *host*, trzeci jako *port* , a na końcu jako *kanał*.

Więcej informacji na ten temat zawiera sekcja Plik MFT command.properties.

Za pomocą właściwości **com.ibm.wmqfte.propertySet** można określić, który plik command.properties ma być używany. Więcej informacji na ten temat zawiera sekcja com.ibm.wmqfte.propertySet.

Jeśli atrybut cmdqm nie jest używany, zadanie domyślnie używa właściwości com.ibm.wmqfte.ant.commandQueueManager , jeśli ta właściwość jest ustawiona. Jeśli właściwość com.ibm.wmqfte.ant.commandQueueManager nie jest ustawiona, podejmowana jest próba nawiązania połączenia z domyślnym menedżerem kolejek zdefiniowanym w pliku command.properties . Format właściwości com.ibm.wmqfte.ant.commandQueueManager jest taki sam, jak atrybut cmdqm , czyli qmgrname@host@port@channel.

rcproperty (właściwość rc)
Wymagane. Określa nazwę właściwości, w której ma zostać zapisany kod powrotu operacji **ping** .

limit czasu
Opcjonalne. Maksymalny czas (w sekundach) oczekiwania zadania na odpowiedź agenta. Minimalny limit czasu wynosi zero sekund, jednak można również określić limit czasu minus jeden, aby komenda

oczekiwał na odpowiedź agenta w nieskończoność. Jeśli dla parametru `timeout` nie zostanie podana żadna wartość, wartością domyślną jest oczekiwanie do 5 sekund na odpowiedź agenta.

Przykład

Ten przykład wysyła żądanie **fte:ping** do serwera `agent1` obsługiwanego przez `qm1`. Żądanie **fte:ping** oczekuje 15 sekund na odpowiedź agenta. Wynik żądania **fte:ping** jest przechowywany we właściwości o nazwie `ping.rc`.

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

Kody powrotu

0
Wykonanie komendy zakończyło się pomyślnie.

2
Przekroczono limit czasu komendy.

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

Zadanie fte: uuid Ant

Generuje pseudolosowy unikalny identyfikator i przypisuje go do danej właściwości. Na przykład można użyć tego identyfikatora do wygenerowania nazw zadań dla innych operacji przesyłania plików.

Atrybuty

Długość
Wymagane. Liczbowa długość identyfikatora UUID do wygenerowania. Ta wartość długości nie obejmuje długości żadnego przedrostka, określonego przez parametr **prefix** .

właściwość
Wymagane. Nazwa właściwości, do której ma zostać przypisany wygenerowany identyfikator UUID.

przedrostek
Opcjonalne. Przedrostek, który ma zostać dodany do wygenerowanego identyfikatora UUID. Ten przedrostek nie jest liczony jako część długości identyfikatora UUID określonego przez parametr **length** .

Przykład

Ten przykład definiuje identyfikator UUID rozpoczynający się od liter ABC , po których następuje 16 pseudolosowych znaków szesnastkowych. Identyfikator UUID jest przypisany do właściwości o nazwie `uuid.property`.

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

fte: filespec Ant element zagnieżdżony

Parametr **fte:filespec** jest używany jako element zagnieżdżony w innych zadaniach. Parametr **fte:filespec** służy do opisania odwzorowania między jednym lub większą liczbą plików źródłowych, katalogów z/OS lub zestawów danycha miejscem docelowym. Zwykle ten element jest używany podczas wyrażania zestawu plików lub katalogów z/OS lub zestawów danych do przeniesienia lub skopiowania.

Zagnieżdżone przez:

- Zadanie fte: filecopy
- Zadanie fte: filemove

Atrybuty specyfikacji źródła

Należy podać jedną z wartości: srcfilespec lub srcqueue.

specyfikacja_pliku_źródł
Określa źródło operacji na pliku. Wartość tego atrybutu może zawierać znak wieloznaczny.

srcqueue (kolejka)
Określa, że źródłem przesyłania jest kolejka. Operacja przesyłania przenosi dane z komunikatów przechowywanych w kolejce określonej przez ten atrybut. Nie można określić tego atrybutu, jeśli zadanie **fte:filespec** jest zagnieżdżone w zadaniu **fte:filecopy** .

Atrybut srcqueue nie jest obsługiwany, gdy agent źródłowy jest agentem mostu protokołu.

Atrybuty specyfikacji miejsca docelowego

Należy podać jedną z wartości: dstdir, dstds, dstfilespace, dstfile, dstqueue lub dstpds.

katalog_dst
Określa katalog jako miejsce docelowe dla operacji na pliku.

z/OS **dstds (dstds)**
Określa zestaw danych jako miejsce docelowe dla operacji na pliku.

Ten atrybut jest obsługiwany tylko wtedy, gdy agent docelowy jest uruchomiony na platformie z/OS .

plik_dst
Określa plik jako miejsce docelowe dla operacji na pliku.

obszar plików dstfilespace
Określa obszar plików jako miejsce docelowe dla operacji na pliku.

Ten atrybut ma zastosowanie tylko wtedy, gdy agent docelowy jest agentem WWW IBM MQ 8.0 , który ma dostęp do obszaru plików bramy WWW.

z/OS **dstpds (dstpds)**
Określa partycjonowany zestaw danych jako miejsce docelowe dla operacji na pliku.

Ten atrybut jest obsługiwany tylko wtedy, gdy agent docelowy jest uruchomiony na platformie z/OS .

kolejka_dst

Określa kolejkę jako miejsce docelowe dla operacji przesyłania pliku do komunikatu. Opcjonalnie można dołączyć nazwę menedżera kolejek do tej specyfikacji, używając formatu QUEUE@QUEUEMANAGER. . Jeśli nazwa menedżera kolejek nie zostanie określona, menedżer kolejek agenta docelowego będzie używany, jeśli właściwość agenta wyjściowego enableClusterQueueInputnie zostanie ustawiona na wartość true. Jeśli właściwość wyjściowa enableClusterQueueInputma wartość true, agent docelowy używa standardowych procedur IBM MQ do określenia położenia kolejki. Należy podać poprawną nazwę kolejki, która istnieje w menedżerze kolejek.

Jeśli zostanie podany atrybut dstqueue , nie można podać atrybutów srcqueue , ponieważ wykluczają się one wzajemnie.

Atrybut dstqueue nie jest obsługiwany, gdy agent docelowy jest agentem mostu protokołu.

Atrybuty opcji źródła

kodowanie (srcencoding)
Opcjonalne. Kodowanie zestawu znaków używane przez plik do przesyłania.

Ten atrybut można określić tylko wtedy, gdy atrybut conversion ma wartość `text.` .

Jeśli nie zostanie podany atrybut srcencoding , do przesyłania tekstu będzie używany zestaw znaków systemu źródłowego.

srceol (srceol)
Opcjonalne. Separator końca wiersza używany przez przesyłany plik. Poprawne wartości to:

- CRLF -znak powrotu karetki, po którym następuje znak nowego wiersza jako ogranicznik końca wiersza. Ta konwencja jest typowa dla systemów Windows .
- LF -znak nowego wiersza jest używany jako separator końca wiersza. Ta konwencja jest typowa dla systemów UNIX .

Atrybut ten można określić tylko wtedy, gdy atrybut conversion ma wartość `text`. Jeśli atrybut srceol nie zostanie podany, przesyłanie tekstowe automatycznie określi poprawną wartość na podstawie systemu operacyjnego agenta źródłowego.

z/OS **srckeeptrailingspaces**
Opcjonalne. Określa, czy końcowe spacje są przechowywane w rekordach źródłowych odczytywanych z zestawu danych w formacie o stałej długości w ramach przesyłania w trybie tekstowym. Poprawne wartości to:

- `true` -spacje na końcu są zachowywane.
- `false` -spacje na końcu są usuwane.

Jeśli atrybut srckeeptrailingspaces nie zostanie określony, wartością domyślną jest `false` .

Ten atrybut można podać tylko wtedy, gdy określono również atrybut srcfilespec i ustawiono atrybut conversion na wartość `text.` .

srcmsgdelimbytes
Opcjonalne. Określa co najmniej jedną wartość bajtową, która ma zostać wstawiona jako separator podczas dodawania wielu komunikatów do pliku binarnego. Każda wartość musi być określona w postaci dwóch cyfr szesnastkowych z zakresu 00-FF, poprzedzonych znakiem x. Wiele bajtów należy oddzielać przecinkami. Na przykład: `srcmsgdelimbytes="x08,xA4"`. Atrybut srcmsgdelimbytes można określić tylko wtedy, gdy określono również atrybut srcqueue . Nie można określić atrybutu srcmsgdelimbytes , jeśli określono również wartość `text` dla atrybutu conversion .

srcmsgdelimtext (tekst komunikatu)
Opcjonalne. Określa sekwencję tekstu, który ma być wstawiany jako separator podczas dodawania wielu komunikatów do pliku tekstowego. Do ogranicznika można dołączyć sekwencje o zmienionym znaczeniu Java dla literałów łańcuchowych. Na przykład: `srcmsgdelimtext="\u007d\n"`. Separator tekstu jest wstawiany przez agenta źródłowego po każdym komunikacie. Separator tekstu jest kodowany do formatu binarnego przy użyciu kodowania źródłowego przesyłania. Każdy komunikat jest odczytywany w formacie binarnym, zakodowany separator jest dodawany do komunikatu w formacie binarnym, a wynik jest przesyłany do agenta docelowego w formacie binarnym. Jeśli strona kodowa agenta źródłowego zawiera stany shift-in i shift-out, agent przyjmuje, że każdy komunikat jest w stanie shift-out na końcu komunikatu. W agencie docelowym dane binarne są przekształcane w taki sam sposób, jak plik do przesyłania tekstowego. Atrybut srcmsgdelimtext można określić tylko wtedy, gdy określono również atrybut srcqueue i wartość `text` dla atrybutu conversion .

srcmsgdelimposition (właściwość srcmsgdelimposition)
Opcjonalne. Określa pozycję, do której wstawiany jest tekst lub separator binarny. Poprawne wartości to:

- `prefix` -separatory są wstawiane do pliku docelowego przed danymi z każdego komunikatu.
- `postfix` -separatory są wstawiane do pliku docelowego po danych z każdego komunikatu.

Atrybut srcmsgdelimposition można określić tylko wtedy, gdy określono również jeden z atrybutów srcmsgdelimbytes lub srcmsgdelimtext .

srcmsggroups (grupy komunikatów)
Opcjonalne. Określa, że komunikaty są grupowane według identyfikatora grupy IBM MQ . Pierwsza kompletna grupa jest zapisywana w pliku docelowym. Jeśli ten atrybut nie jest określony, wszystkie komunikaty w kolejce źródłowej są zapisywane w pliku docelowym. Atrybut srcmsggroups można określić tylko wtedy, gdy określono również atrybut srcqueue .

srcqueuetimeout (limit czasu kolejki)
Opcjonalne. Określa czas (w sekundach) oczekiwania na spełnienie jednego z następujących warunków:

- Dla nowego komunikatu, który ma zostać zapisany w kolejce.
- Jeśli określono atrybut srcmsggroups , pełna grupa ma zostać zapisana w kolejce.

Jeśli żaden z tych warunków nie zostanie spełniony w czasie określonym przez wartość srcqueuetimeout, agent źródłowy zatrzymuje odczyt z kolejki i kończy przesyłanie. Jeśli atrybut srcqueuetimeout nie zostanie określony, agent źródłowy natychmiast zatrzymuje odczyt z kolejki źródłowej, jeśli kolejka źródłowa jest pusta lub, w przypadku określenia atrybutu srcmsggroups , jeśli w kolejce nie ma pełnej grupy. Atrybut srcqueuetimeout można określić tylko wtedy, gdy określono również atrybut srcqueue .

Informacje na temat ustawiania wartości *srcqueuetimeout* zawiera sekcja Wskazówki dotyczące określania czasu oczekiwania na przesłanie komunikatu do pliku.

z/OS **srcrecdelimbytes,**
Opcjonalne. Określa co najmniej jedną wartość bajtową, która ma być wstawiona jako separator podczas dodawania wielu rekordów ze zbioru źródłowego zorientowanego na rekordy do pliku binarnego. Każdą wartość należy podać jako dwie cyfry szesnastkowe z zakresu 00-FF z przedrostkiem x. Wiele bajtów należy oddzielać przecinkami. Na przykład:

```
srcrecdelimbytes="x08,xA4"
```

Atrybut srcrecdelimbytes można określić tylko wtedy, gdy plik źródłowy przesyłania jest zorientowany na rekordy, na przykład zestaw danych z/OS , a plik docelowy jest zwykłym plikiem niezorientowanym na rekordy. Nie można określić atrybutu srcrecdelimbytes , jeśli określono również wartość `text` dla atrybutu conversion .

srcrecdelimpos,
Opcjonalne. Określa pozycję, do której wstawiany jest separator binarny. Poprawne wartości to:

- prefiks-separatory są wstawiane do pliku docelowego przed danymi z każdego rekordu pliku źródłowego zorientowanego na rekordy.
- postfix-separatory są wstawiane do pliku docelowego po danych z każdego rekordu pliku źródłowego zorientowanego na rekordy.

Atrybut srcrecdelimpos można określić tylko wtedy, gdy określono również atrybut srcrecdelimbytes .

Atrybuty opcji docelowej

kodowanie_dst
Opcjonalne. Kodowanie zestawu znaków, które ma być używane dla przesyłanego pliku.

Ten atrybut można określić tylko wtedy, gdy atrybut conversion ma wartość `text.` .

Jeśli atrybut dstencoding nie jest określony, do przesyłania tekstu używany jest zestaw znaków systemu docelowego.

dsteol (dsteol)
Opcjonalne. Separator końca wiersza, który ma być używany dla przesyłanego pliku. Poprawne wartości to:

- CRLF -znak powrotu karetki, po którym następuje znak nowego wiersza jako ogranicznik końca wiersza. Ta konwencja jest typowa dla systemów Windows .

- LF -znak nowego wiersza jest używany jako separator końca wiersza. Ta konwencja jest typowa dla systemów UNIX .

Ten atrybut można określić tylko wtedy, gdy atrybut conversion ma wartość `text.` .

Jeśli atrybut dsteol nie zostanie podany, przesyłanie tekstowe automatycznie określi poprawną wartość na podstawie systemu operacyjnego agenta docelowego.

dstmsgdelimbytes (dstmsgdelimbajty)
Opcjonalne. Określa separator szesnastkowy, który ma być używany podczas dzielenia pliku binarnego na wiele komunikatów. Wszystkie komunikaty mają ten sam identyfikator grupy IBM MQ ; ostatni komunikat w grupie ma ustawioną flagę IBM MQ LAST_MSG_IN_GROUP. Format określania bajtu szesnastkowego jako separatora to `xNN`, gdzie N jest znakiem z zakresu od 0 do 9 lub od a do f. Można określić sekwencję bajtów szesnastkowych jako separator, podając rozdzielaną przecinkami listę bajtów szesnastkowych, na przykład: `x3e,x20,x20,xbf`.

Atrybut dstmsgdelimbytes można określić tylko wtedy, gdy określono również atrybut dstqueue , a przesyłanie odbywa się w trybie binarnym. Można określić tylko jeden z atrybutów dstmsgsize, dstmsgdelimbytesi dstmsgdelimpattern .

dstmsgdelimpattern
Opcjonalne. Określa wyrażenie regularne Java , które ma być używane podczas dzielenia pliku tekstowego na wiele komunikatów. Wszystkie komunikaty mają ten sam identyfikator grupy IBM MQ ; ostatni komunikat w grupie ma ustawioną flagę IBM MQ LAST_MSG_IN_GROUP. Format określania wyrażenia regularnego jako separatora jest wyrażeniem regularnym ujętym w nawias, *(regular_expression)* lub ujętym w cudzysłów, "*regular_expression*". Więcej informacji na ten temat zawiera sekcja Wyrażenia regularne używane przez produkt MFT.

Domyślnie długość łańcucha, który może być zgodny z wyrażeniem regularnym, jest ograniczona przez agenta docelowego do pięciu znaków. To zachowanie można zmienić za pomocą właściwości agenta **maxDelimiterMatchLength** . Więcej informacji na ten temat zawiera sekcja Zaawansowane właściwości agenta MFT.

Atrybut dstmsgdelimpattern można określić tylko wtedy, gdy określono również atrybut dstqueue , a przesyłanie odbywa się w trybie tekstowym. Można określić tylko jeden z atrybutów dstmsgsize, dstmsgdelimbytesi dstmsgdelimpattern .

dstmsgdelimposition (pozycja dstmsgdelimposition)
Opcjonalne. Określa pozycję, w której powinien się znajdować ogranicznik tekstu lub separator binarny. Poprawne wartości to:

- `prefix` -separatory są oczekiwane na początku każdego wiersza.
- `postfix` -separatory są oczekiwane na końcu każdego wiersza.

Atrybut dstmsgdelimposition można określić tylko wtedy, gdy określono również atrybut dstmsgdelimpattern .

dstmsgincludedelim (dstmsgincludedelim)
Opcjonalne. Określa, czy dołączyć separator używany do podziału pliku na wiele komunikatów w komunikatach. Jeśli określono atrybut dstmsgincludedelim , separator jest dołączany na końcu komunikatu zawierającego dane pliku poprzedzające separator. Domyślnie separator nie jest uwzględniany w komunikatach. Atrybut dstmsgincludedelim można określić tylko wtedy, gdy określono również jeden z atrybutów dstmsgdelimpattern i dstmsgdelimbytes .

dstmsgpersist (dstmsgpersist)
Opcjonalne. Określa, czy komunikaty zapisywane w kolejce docelowej są trwałe. Poprawne wartości to:

- `true` -zapisanie trwałych komunikatów w kolejce docelowej. Jest to wartość domyślna.
- `false` -zapisanie nietrwałych komunikatów w kolejce docelowej.
- `qdef` -wartość trwałości jest pobierana z atrybutu DefPersistence kolejki docelowej.

Atrybut ten można podać tylko wtedy, gdy określono również atrybut dstqueue .

dstmsgprops,
Opcjonalne. Określa, czy pierwszy komunikat zapisany w kolejce docelowej przez operację przesyłania ma ustawione właściwości komunikatu IBM MQ . Dozwolone są następujące wartości:

- `true` -ustaw właściwości pierwszego komunikatu utworzonego przez operację przesyłania.
- `false` -nie ustawiaj właściwości komunikatu dla pierwszego komunikatu utworzonego podczas przesyłania. Jest to wartość domyślna.

Więcej informacji na ten temat zawiera sekcja Właściwości komunikatów produktuMQ ustawiane przez produkt MFT w przypadku komunikatów zapisywanych do kolejek docelowych.

Atrybut ten można podać tylko wtedy, gdy określono również atrybut dstqueue .

dstmsgsize
Opcjonalne. Określa, czy plik ma być dzielony na wiele komunikatów o stałej długości. Wszystkie komunikaty mają ten sam identyfikator grupy IBM MQ ; ostatni komunikat w grupie ma ustawioną flagę IBM MQ LAST_MSG_IN_GROUP. Wielkość komunikatów jest określana przez wartość dstmsgsize. Format parametru dstmsgsize to *długośćjednostki*, gdzie *długość* jest dodatnią liczbą całkowitą, a *jednostki* jest jedną z następujących wartości:

- B -bajty. Minimalna dozwolona wartość to dwa razy więcej niż maksymalna liczba bajtów na znak strony kodowej komunikatów docelowych.
- K -Kibibajty. Odpowiada to 1024 bajtom.
- M -mebibajty. Odpowiada to 1024 kibibajtom.

Jeśli plik jest przesyłany w trybie tekstowym i znajduje się w zestawie znaków dwubajtowych lub w zestawie znaków wielobajtowych, plik jest dzielony na komunikaty znajdujące się na najbliższej granicy znaku w odniesieniu do określonej wielkości komunikatu.

Atrybut dstmsgsize można określić tylko wtedy, gdy określono również atrybut dstqueue . Można określić tylko jeden z atrybutów dstmsgsize, dstmsgdelimbytesi dstmsgdelimpattern .

dstunsupportedcodepage (dstunsupportedcodepage)
Opcjonalne. Określa działanie, które ma zostać wykonane, jeśli docelowy menedżer kolejek określony przez atrybut dstqueue nie obsługuje strony kodowej używanej podczas przesyłania danych pliku do kolejki w ramach przesyłania tekstowego. Poprawne wartości tego atrybutu są następujące:

- `binary` -kontynuuj przesyłanie, ale nie stosuj konwersji strony kodowej dla przesyłanych danych. Podanie tej wartości jest równoważne ustawieniu atrybutu konwersji na wartość `text`.
- `fail` -nie kontynuuj operacji przesyłania. Plik jest rejestrowany jako plik, którego przesłanie nie powiodło się. Jest to opcja domyślna.

Atrybut dstunsupportedcodepage można określić tylko wtedy, gdy określono również atrybut dstqueue i wartość `text` dla atrybutu conversion .

z/OS **dsttruncaterecords (dsttruncaterecords)**
Opcjonalne. Określa, że rekordy docelowe dłuższe niż atrybut zestawu danych LRECL są obcinane. Jeśli ma wartość true, rekordy są obcinane. Jeśli ma wartość false, rekordy są zawijane. Wartość domyślna to false. Ten parametr jest poprawny tylko w przypadku przesyłania w trybie tekstowym, gdy miejscem docelowym jest zestaw danych.

Inne atrybuty

Suma kontrolna
Opcjonalne. Określa algorytm używany do sprawdzania sumy kontrolnej przesłanych plików.

- `MD5` -użyj algorytmu kodowania mieszającego MD5 .
- `NONE` -nie używaj algorytmu sumy kontrolnej.

Jeśli atrybut checksum (suma kontrolna) nie zostanie podany, zostanie użyta wartość domyślna `MD5` .

konwersja
Opcjonalne. Określa typ konwersji, który ma być zastosowany do zbioru podczas jego przesyłania. Dozwolone są następujące wartości:

- `binary` -nie stosuj konwersji.
- `text` -stosuje konwersję strony kodowej między systemem źródłowym i docelowym. Należy również zastosować konwersję ograniczników wierszy. Atrybuty `srcencoding`, `dstencoding`, `srceol` i `dsteol` mają wpływ na stosowaną konwersję.

Jeśli atrybut conversion nie zostanie określony, zostanie podana wartość domyślna `binary` .

Nadpisanie
Opcjonalne. Określa, czy istniejący plik docelowy z/OS lub zestaw danych może zostać nadpisany przez operację. Jeśli zostanie podana wartość `true`, wszystkie istniejące pliki docelowe z/OS lub zestawy danych zostaną nadpisane. Jeśli zostanie podana wartość `false`, istnienie zduplikowanego pliku z/OS lub zestawu danych w miejscu docelowym spowoduje niepowodzenie operacji. Jeśli atrybut overwrite nie zostanie podany, zostanie podana wartość domyślna `false` .

rekurencyjne
Opcjonalne. Określa, czy operacja przesyłania pliku jest rekurencyjna w podkatalogach. Jeśli zostanie podana wartość `true`, operacja przesyłania jest rekurencyjna w podkatalogach. Jeśli zostanie podana wartość `false`, operacja przesyłania nie będzie rekurencyjna w podkatalogach. Jeśli atrybut recurse nie jest określony, zostanie podana wartość domyślna `false` .

Przykład

W tym przykładzie określono fte: filespec z plikiem źródłowym `file1.bin` i plikiem docelowym `file2.bin.` .

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

Zadania pokrewne
Używanie produktu Apache Ant z produktem MFT

fte: metadane Ant zagnieżdżone elementy

Metadane są używane do przenoszenia dodatkowych informacji zdefiniowanych przez użytkownika za pomocą operacji przesyłania plików.

Więcej informacji na temat sposobu używania metadanych przez produkt Managed File Transfer zawiera sekcja “Metadane dla programów zewnętrznych MFT” na stronie 2190 .

Zagnieżdżone przez:

- Zadanie fte: filecopy
- Zadanie fte: filemove
- Zadanie fte: call

Parametry określone jako elementy zagnieżdżone

fte: pozycja
Należy określić co najmniej jedną pozycję wewnątrz elementu zagnieżdżonego fte:metadata . Można określić więcej niż jedną pozycję. Pozycje wiążą nazwę klucza z wartością. Klucze muszą być unikalne w bloku fte:metadata

Atrybuty pozycji

Nazwa
Wymagane. Nazwa klucza należącego do tej pozycji. Ta nazwa musi być unikalna wśród wszystkich parametrów **entry** zagnieżdżonych w elemencie fte: metadata .

Wartość
Wymagane. Wartość, która ma zostać przypisana do tej pozycji.

Przykład

W tym przykładzie przedstawiono definicję fte:metadata , która zawiera dwie pozycje.

```
<fte:metadata>
    <fte:entry name="org.foo.partColor" value="red"/>
    <fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

Zadania pokrewne

Używanie produktu Apache Ant z produktem MFT

Zagnieżdżone elementy wywołania programu

Programy można uruchamiać przy użyciu jednego z pięciu zagnieżdżonych elementów: `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrci` `fte:command`. Te zagnieżdżone elementy nakazają agentowi wywołanie programu zewnętrznego w ramach jego przetwarzania. Przed uruchomieniem programu należy się upewnić, że komenda znajduje się w położeniu określonym przez właściwość commandPath w pliku `agent.properties` agenta, który uruchomiła komendę.

Mimo że każdy element wywołania programu ma inną nazwę, współużytkują ten sam zestaw atrybutów i ten sam zestaw zagnieżdżonych elementów. Programy mogą być uruchamiane przez zadania Ant **fte:filecopy**, **fte:filemove**i **fte:command** .

Nie można wywoływać programów z agenta mostu Connect:Direct .

Zadania Ant, które mogą wywoływać programy:

- Zadanie fte: filecopy zagnieżdża parametry wywołania programu przy użyciu elementów zagnieżdżonych `fte:predst`, `fte:postdst`, `fte:presrci` `fte:postsrc` .
- Zadanie fte: filemove zagnieżdża parametry wywołania programu przy użyciu elementów zagnieżdżonych `fte:predst`, `fte:postdst`, `fte:presrci` `fte:postsrc` .
- Zadanie fte: call zagnieżdża parametry wywołania programu przy użyciu zagnieżdżonego elementu `fte:command` .

Atrybuty

komenda
Wymagane. Określa nazwę programu, który ma zostać wywołany. Aby agent mógł uruchomić komendę, komenda musi znajdować się w położeniu określonym przez właściwość commandPath w pliku `agent.properties` agenta. Więcej informacji na ten temat zawiera sekcja commandPath MFT właściwość. Wszystkie informacje o ścieżce określone w atrybucie `command` są traktowane jako względne w stosunku do położenia określonego przez właściwość commandPath . Jeśli parametr `type` ma wartość `executable`, oczekiwany jest program wykonywalny, w przeciwnym razie oczekiwany jest skrypt odpowiedni dla typu wywołania.

liczność ponowień
Opcjonalne. Liczba ponownych prób wywołania programu, jeśli program nie zwraca kodu powrotu oznaczającego powodzenie. Program nazwany przez atrybut `command` jest wywoływany do tej liczby razy. Wartość przypisana do tego atrybutu musi być nieujemna. Jeśli atrybut `retrycount` nie zostanie określony, zostanie użyta wartość domyślna wynosząca zero.

ponowna próba oczekiwania
Opcjonalne. Czas oczekiwania (w sekundach) przed ponowną próbą wywołania programu. Jeśli program o nazwie określonej przez atrybut `command` nie zwraca kodu powrotu powodzeń, a atrybut `retrycount` określa wartość niezerową, ten parametr określa czas oczekiwania między ponownymi próbami. Wartość przypisana do tego atrybutu musi być nieujemna. Jeśli atrybut `retrywait` nie zostanie określony, zostanie użyta wartość domyślna wynosząca zero.

pomyślna
Opcjonalne. Wartość tego atrybutu jest używana do określenia, kiedy wywołanie programu zakończyło się pomyślnie. Kod powrotu procesu dla komendy jest wartościowany przy użyciu tego wyrażenia. Wartość może składać się z jednego lub większej liczby wyrażeń połączonych znakiem pionowej kreski (|) w celu oznaczenia wartości boolowskiej ORlub znaku ampersand (&) Znak oznaczający wartość boolowską AND. Każde wyrażenie może mieć jeden z następujących typów:

- Liczba wskazująca test równości między kodem powrotu procesu a liczbą.
- Liczba z przedrostkiem ">" oznaczającym test większy niż między liczbą a kodem powrotu procesu.
- Liczba z przedrostkiem "<" oznaczającym test mniejszości między liczbą a kodem powrotu procesu.
- Liczba z przedrostkiem "!" Znak oznaczający nierówny test między liczbą a kodem powrotu procesu.

Na przykład: >2&<7&!5|0|14 jest interpretowany jako następujące kody powrotu zakończone powodzeniem: 0, 3, 4, 6, 14. Wszystkie inne kody powrotu są interpretowane jako nieudane. Jeśli atrybut `successrc` nie zostanie określony, zostanie użyta wartość domyślna wynosząca zero. Oznacza to, że komenda jest oceniana jako pomyślnie uruchomiona, jeśli i tylko wtedy, gdy zwraca kod o wartości zero.

typ
Opcjonalne. Wartość tego atrybutu określa typ wywoływanego programu. Podaj jedną z następujących opcji:

executable
Zadanie wywołuje program wykonywalny. Mogą mieć dodatkowe argumenty określone za pomocą zagnieżdżonego elementu arg . Program powinien być dostępny w ścieżce commandPath i tam, gdzie ma to zastosowanie, mieć ustawione uprawnienie do wykonywania. Skrypty UNIX mogą być wywoływane pod warunkiem, że określają program powłoki (na przykład pierwszy wiersz pliku skryptu powłoki to: `#!/bin/sh`). Dane wyjściowe komendy zapisywane do wyjścia standardowego wyjścia błędów (stderr) lub wyjścia standardowego (stdout) są wysyłane do dziennika Managed File Transfer dla wywołania. Jednak ilość danych wyjściowych jest ograniczona przez konfigurację agenta. Wartością domyślną jest 10K bajtów danych, ale można ją nadpisać za pomocą właściwości agenta: maxCommandOutput.

Skrypt przeciwny
Zadanie uruchamia określony skrypt Ant za pomocą komendy **`fteAnt`** . Właściwości można określić przy użyciu zagnieżdżonego elementu property . Elementy docelowe Ant można określić przy użyciu zagnieżdżonego elementu target . Oczekuje się, że skrypt Ant będzie dostępny w ścieżce commandPath. Dane wyjściowe komendy Ant zapisywane do standardowego wyjścia błędów (stderr) lub wyjścia standardowego (stdout) są wysyłane do dziennika Managed File Transfer dla wywołania. Jednak ilość danych wyjściowych jest ograniczona przez konfigurację agenta. Wartością domyślną jest 10K bajtów danych, ale można nadpisać tę wartość domyślną, używając właściwości agenta: maxCommand.

z/OS **JCL**
Wartość `jcl` jest obsługiwana tylko w systemie z/OS i uruchamia określony skrypt JCL z/OS . Kod JCL jest wprowadzany jako zadanie i wymaga obecności karty pracy. Po pomyślnym wprowadzeniu zadania dane wyjściowe komendy JCL zapisane w dzienniku Managed File Transfer zawierają następujący tekst: JOB *nazwa_zadania*(*id_zadania*), gdzie:

- *nazwa_zadania* to nazwa zadania identyfikowana przez kartę zadania w JCL.
- *id_zadania* to identyfikator zadania wygenerowany przez system z/OS .

Jeśli zadanie nie może zostać pomyślnie wprowadzone, wykonanie komendy skryptu JCL nie powiedzie się i w dzienniku zostanie zapisany komunikat wskazujący przyczynę niepowodzenia

(na przykład brak karty zadania). Aby zrozumieć, czy zadanie zostało wykonane pomyślnie, należy użyć usługi systemowej, takiej jak SDSF. Managed File Transfer nie udostępnia informacji, ponieważ wprowadza tylko zadanie; system następnie określa, kiedy uruchomić zadanie i w jaki sposób prezentowane są dane wyjściowe zadania. Ponieważ skrypt JCL jest wprowadzany jako zadanie wsadowe, nie zaleca się określania parametru `jcl` dla elementu zagnieżdżonego `presrc` lub `predst` , ponieważ wiadomo tylko, że zadanie zostało pomyślnie wprowadzone, a nie, że zostało pomyślnie wykonane przed rozpoczęciem przesyłania. Brak zagnieżdżonych elementów, które są poprawne dla typu `jcl`.

W poniższym przykładzie przedstawiono zadanie JCL:

```
//MYJOB JOB
//*
//MYJOB EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=H
//SYSUT1 DD DSN=FRED.DEMO.TXT,DISP=SHR
//SYSUT2 DD DSN=BOB.DEMO.TXT,DISP=(NEW,CATLG),
// RECFM=VB,LRECL=133,BLKSIZE=2048,
// SPACE=(TRK,(30,5),RLSE)
//SYSIN DD DUMMY
```

Parametry określone jako elementy zagnieżdżone

fte: arg
Poprawne tylko wtedy, gdy wartością atrybutu `type` jest executable. Zagnieżdżone elementy `fte:arg` służą do określania argumentów programu, który jest wywoływany jako część wywołania programu. Argumenty programu są budowane na podstawie wartości określonych przez elementy `fte:arg` w kolejności, w jakiej występują elementy `fte:arg` . Można określić zero lub więcej elementów `fte:arg` jako zagnieżdżone elementy wywołania programu.

fte: właściwość
Poprawna tylko wtedy, gdy wartością atrybutu `type` jest antscript. Atrybuty `name` i `value` zagnieżdżonych elementów `fte:property` służą do przekazywania par nazwa-wartość do skryptu Ant . Można określić zero lub więcej elementów `fte:property` jako zagnieżdżone elementy wywołania programu.

fte: cel
Poprawna tylko wtedy, gdy wartością atrybutu `type` jest antscript. Określ cel w skrypcie Ant do wywołania. Można określić zero lub więcej elementów `fte:target` jako zagnieżdżone elementy wywołania programu.

Arg-atrybuty

wartość
Wymagane. Wartość argumentu, który ma zostać przekazany do wywoływanego programu.

Atrybuty elementu Property

nazwa
Wymagane. Nazwa właściwości, która ma zostać przekazana do skryptu Ant .

wartość
Wymagane. Wartość, która ma zostać powiązana z nazwą właściwości przekazywaną do skryptu Ant .

Przykłady

W tym przykładzie przedstawiono wywołanie programu `fte:postsrc` określone jako część zadania fte: filecopy . Wywołanie programu jest przeznaczone dla programu o nazwie `post.sh` i jest dostarczane z pojedynczym argumentem `/home/fteuser2/file.bin.` .

```
<fte:filecopy
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
 src="agent1@qm1" dst="agent2@qm2"
```

```
 rcproperty="copy.result">
<fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>
<fte:postsrc command="post.sh" successrc="1" >
    <fte:arg value="/home/fteuser2/file.bin"/>
</fte:postsrc>
</fte:filecopy>
```

W tym przykładzie przedstawiono wywołanie programu `fte:command` określone jako część zadania fte: call . Wywołanie programu jest przeznaczone dla pliku wykonywalnego o nazwie `command.sh`, do którego nie są przekazywane żadne argumenty wiersza komend. Jeśli program `command.sh` nie zwróci kodu powrotu 1, komenda zostanie ponowiona po upływie 30 sekund.

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
 agent="agent1@qm1"
 rcproperty="call.rc"
 origuser="bob"
 jobname="${job.id}">
<fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30"/>
</fte:call>
```

W tym przykładzie przedstawiono wywołanie programu `fte:command` określone jako część zadania fte: call . Wywołanie programu jest przeznaczone dla celów kopiowania i kompresji w skrypcie Ant o nazwie `script.xml`, do którego są przekazywane dwie właściwości.

```
<fte:call
cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
 agent="agent1@qm1"
 rcproperty="call.rc"
 origuser="bob"
 jobname="${job.id}">
<fte:command command="script.xml" type="antscript">
  <property name="src" value="AGENT5@QM5"/>
  <property name="dst" value="AGENT3@QM3"/>
  <target name="copy"/>
  <target name="compress"/>
</fte:command>
</fte:call>
```

Zadania pokrewne

Określanie programów, które mają być uruchamiane z systemem MFT

Używanie produktu Apache Ant z produktem MFT

Procedury zewnętrzne MFT dla odwołania do dostosowywania

Informacje uzupełniające, które ułatwiają konfigurowanie procedur zewnętrznych dla produktu Managed File Transfer.

Pojęcia pokrewne

Źródłowe i docelowe procedury zewnętrzne MFT

Metadane dla programów zewnętrznych MFT

Istnieją trzy różne typy metadanych, które można dostarczyć do procedur zewnętrznych dla produktu Managed File Transfer: środowisko, przesyłanie i metadane plików. Te metadane są prezentowane jako odwzorowania par klucz-wartość Java .

Metadane środowiska

Metadane środowiska są przekazywane do wszystkich procedur zewnętrznych i opisują środowisko wykonawcze agenta, z którego jest wywoływana procedura zewnętrzna. Te metadane są tylko do odczytu i nie mogą być aktualizowane przez żadną procedurę zewnętrzną.

Tabela 882. Metadane środowiska	
Klucz	**Opis**
AGENT_CONFIGURATION_DIRECTORY_KEY,	Nazwa katalogu zawierającego informacje o konfiguracji agenta.
AGENT_PRODUCT_DIRECTORY_KEY,	Nazwa katalogu, w którym został zainstalowany kod agenta.
KLUCZ_WERSJI_AGENTA	Numer wersji środowiska wykonawczego agenta, które wywołuje procedurę zewnętrzną.

Nazwy kluczy i wartości podane w tabeli 1 są stałymi zdefiniowanymi w interfejsie EnvironmentMetaDataConstants .

Metadane przesyłania

Metadane przesyłania są przekazywane do wszystkich procedur zewnętrznych. Metadane składają się z wartości dostarczonych przez system i wartości podanych przez użytkownika. Jeśli zostaną zmienione jakiekolwiek wartości systemowe, zmiany te zostaną zignorowane. Początkowe wartości podane przez użytkownika dla źródłowej procedury zewnętrznej rozpoczęcia przesyłania są oparte na wartościach podanych podczas definiowania przesyłania. Agent źródłowy może zmieniać wartości podane przez użytkownika w ramach przetwarzania procedury zewnętrznej rozpoczęcia przesyłania źródła. Ta procedura zewnętrzna jest wywoływana przed rozpoczęciem przesyłania całego pliku. Te zmiany są używane w kolejnych wywołaniach innych procedur obsługi wyjścia, które odnoszą się do tego przesyłania. Metadane przesyłania są stosowane do całej operacji przesyłania.

Mimo że wszystkie procedury zewnętrzne mogą odczytywać wartości z metadanych przesyłania, tylko źródłowa procedura zewnętrzna może zmieniać metadane przesyłania.

Nie można używać metadanych przesyłania do propagowania informacji między różnymi operacjami przesyłania plików.

Metadane przesyłania dostarczone przez system zostały wyszczególnione w tabeli 2:

Tabela 883. Metadane przesyłania	
Klucz	**Opis**
KLUCZ_AGENTA_DOCELOWEGO	Nazwa agenta, który jest miejscem docelowym przesyłania.
KLUCZ_NAZWY_ZADANIA	Nazwa zadania powiązana z żądaniem transferu
MQMD_XX_ENCODE_CASE_ONE klucz_użytkownika	Pole użytkownika MQMD z komunikatu używanego do wprowadzania żądania przesyłania
ORYGINALNY_KLUCZ_HOSTA	Nazwa hosta określona jako nazwa hosta źródłowego w żądaniu przesyłania
ORYGINALNY_KLUCZ_UŻYTKOWNIKA	Nazwa użytkownika określona jako ID użytkownika inicjującego w żądaniu przesyłania
KLUCZ_AGENT_ŹRÓDŁOWY	Nazwa agenta, który jest źródłem operacji przesyłania
KLUCZ_ID_TRANSAKCJI	Identyfikator operacji przesyłania

Nazwy kluczy i wartości podane w tabeli 2 są stałymi, które są zdefiniowane w interfejsieDataConstants TransferMeta.

Metadane pliku

Metadane pliku są przekazywane do źródłowego wyjścia początkowego przesyłania jako część specyfikacji pliku. Istnieją osobne metadane dla plików źródłowych i docelowych.

Nie można używać metadanych pliku do propagowania informacji między różnymi operacjami przesyłania plików.

Tabela 884. Metadane pliku

Klucz	Dozwolone wartości	Opis
CONVERT_LINE_SEPARATORS,		Wartość klucza używana do przesyłania tekstu w celu wskazania, czy sekwencje separatorów wierszy CRLF (znak powrotu karetki-znak nowego wiersza) lub LF (znak nowego wiersza) w danych źródłowych są przekształcane w sekwencję separatorów wierszy w miejscu docelowym.
KLUCZ_DOSTARCZANIA		Wartość klucza używana do definiowania separatora do oddzielania danych rekordu podczas przesyłania danych zorientowanych na rekordy do zwykłych plików. Używany również do przesyłania komunikatów do pliku i przesyłania plików do komunikatu.
DELIMITER_POSITION_KEY,	DELIMITER_POSITION_PREFIX_VALUE DELIMITER_POSITION_POSTFIX_VALUE	W połączeniu z separatorem DELIMITER_KEY służy do definiowania pozycji separatora (przedrostka lub przyrostku).
KLUCZ_TYPU_DOSTARCZANIA	DELIMITER_TYPE_BINARY_VALUE, DELIMITER_TYPE_TEXT_VALUE, DELIMITER_TYPE_SIZE_WARTOŚĆ	W połączeniu z DELIMITER_KEY służy do definiowania typu separatora.
KLUCZ_ISTNIEJĄCY_DOCELOWY	DESTINATION_EXIST_KEY_ERROR_VALUE WARTOŚĆ_DESTINATION_EXIST_KEY_OVERWRITE_VALUE	Określa zachowanie przesyłania plików, jeśli plik docelowy istnieje.
KLUCZ ALIASU PLIKU		Wartość klucza używana do definiowania aliasu dla przesyłanego pliku.
FILE_CHECKSUM_METHOD_KEY (klucz metody sumy kontrolnej pliku)	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	Określa metodę sumy kontrolnej używaną podczas przesyłania pliku.
KLUCZ_KONWERSJI_PLIKU	FILE_CONVERSION_TEXT_VALUE FILE_CONVERSION_BINARY_VALUE	Określa typ konwersji stosowany do zawartości pliku.
FILE_ENCODING_KEY (klucz kodowania plików)		Określa kodowanie używane dla pliku tekstowego.
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE WARTOŚĆ_FILE_END_OF_LINE_CRLF_VALUE	Określa sekwencję znaków oznaczającą koniec wiersza: < LF > lub < CR> < LF>.
ALIAS_OBSZARU_PLIKÓW		Określa alias pliku w obszarze plików. **Uwaga:** Te metadane mogą być używane tylko wtedy, gdy FILE_TYPE_KEY ma wartość FILE_TYPE_FILE_SPACE_VALUE
NAZWA_OBSZARU_PLIKÓW		Określa nazwę obszaru plików. **Uwaga:** Te metadane mogą być używane tylko wtedy, gdy FILE_TYPE_KEY ma wartość FILE_TYPE_FILE_SPACE_VALUE

Tabela 884. Metadane pliku (kontynuacja)

Klucz	Dozwolone wartości	Opis
KLUCZ_TYPU_PLIKU	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	Określa specyfikację pliku docelowego, kolejki lub obszaru plików.
KLUCZ_GRUPY		Wartość klucza używana do przesyłania komunikatów do pliku w celu określenia grupy komunikatów do odczytania z kolejki źródłowej. Ten atrybut jest poprawny tylko wtedy, gdy atrybut USE_GROUPS_KEY ma wartość USE_GROUPS_TRUE_VALUE.
INCLUDE_DELIMITER_IN_MESSAGE_KEY	INCLUDE_DELIMITER_IN_MESSAGE_TRUE_VALUE INCLUDE_DELIMITER_IN_MESSAGE_FALSE_VALUE	Wartość klucza używana do przesyłania plików do komunikatów w celu określenia, czy na końcu komunikatów mają być uwzględniane separatory, które zostały użyte do podzielenia pliku na wiele komunikatów. Ten atrybut jest poprawny tylko wtedy, gdy atrybut DELIMITER_TYPE_KEY ma wartość DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE.
INSERT_RECORD_LINE_SEPARATOR_KEY		Wartość klucza używana do przesyłania tekstu z plików zorientowanych na rekordy w celu określenia, czy separatory wierszy są wstawiane do danych po każdym rekordzie.
KEEP_TRAILING_SPACES_KEY, KLUCZ	KEEP_TRAILING_SPACES_TRUE_VALUE WARTOŚĆ_KEEP_TRAILING_SPACES_FALSE_VALUE	Wartość klucza używana do określenia, czy końcowe spacje są usuwane z rekordów odczytanych z zestawów danych o stałej długości.
NEW_RECORD_ON_LINE_SEPARATOR_KEY		Wartość klucza używana do przesyłania tekstu do plików zorientowanych na rekordy w celu określenia, czy separatory wierszy w danych są dołączane do danych rekordu, czy też powodują powstanie nowego rekordu (i nie są zapisywane).
PERSISTENT_KEY (klucz trwały)	WARTOŚĆ_PRAWDA_TRWAŁOŚCI WARTOŚĆ_TRWANIA_TRWAŁEGO PERSISTENT_QDEF_VALUE (wartość trwałej kolejki)	Wartość klucza używana do przesyłania plików do komunikatów w celu określenia, czy komunikaty są trwałe.
SET_MQ_PROPS_KEY	SET_MQ_PROPS_TRUE_VALUE SET_MQ_PROPS_FALSE_VALUE (wartość proxy)	Wartość klucza używana podczas przesyłania pliku do komunikatu w celu określenia, czy właściwości komunikatu IBM MQ są ustawiane w pierwszym komunikacie w pliku, a także w przypadku wystąpienia błędu w każdym komunikacie zapisanym w kolejce.

Tabela 884. Metadane pliku (kontynuacja)		
Klucz	**Dozwolone wartości**	**Opis**
NIEROZPOZNANY KLUCZ STRONY KODU	NIEROZPOZNANY KOD PAGE_FAIL_VALUE NIEROZPOZNANY KOD PAGE_BINARY_VALUE	Wartość klucza używana do przesyłania plików do komunikatów w celu określenia, czy przesyłanie w trybie tekstowym kończy się niepowodzeniem lub czy wykonywana jest konwersja, jeśli strona kodowa danych nie jest rozpoznawana przez docelowy menedżer kolejek.
USE_GROUPS_KEY	USE_GROUPS_TRUE_VALUE USE_GROUPS_FALSE_VALUE	Wartość klucza używana do przesyłania komunikatów do pliku w celu określenia, czy ma być przesyłana tylko pełna grupa komunikatów z kolejki źródłowej.
KLUCZ_CZASU_OCZEKIWANIA		Wartość klucza używana do przesyłania komunikatów do pliku w celu określenia czasu (w sekundach) oczekiwania agenta źródłowego na jeden z następujących przypadków: • Komunikat, który ma być wyświetlany w kolejce źródłowej, jeśli kolejka jest pusta lub stała się pusta, jeśli wartością parametru USE_GROUPS_KEY jest FALSE. • Pełna grupa, która ma być wyświetlana w kolejce źródłowej, jeśli parametr USE_GROUPS_KEY ma wartość TRUE.

Nazwy kluczy i wartości podane w tabeli 3 są stałymi zdefiniowanymi w interfejsieDataConstants FileMeta.

Pojęcia pokrewne

“Interfejsy Java dla programów zewnętrznych MFT” na stronie 2201
Tematy w tej sekcji zawierają informacje uzupełniające na temat interfejsów Java dla procedur zewnętrznych.

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Procedury zewnętrzne monitora zasobów MFT” na stronie 2194
Procedury zewnętrzne monitora zasobów umożliwiają skonfigurowanie kodu niestandardowego, który ma być uruchamiany po spełnieniu warunku wyzwalacza monitora, zanim zostanie uruchomione powiązane zadanie.

“MFT Właściwości agenta dla procedur zewnętrznych” na stronie 2199
Oprócz właściwości standardowych w pliku `agent.properties` istnieje kilka właściwości zaawansowanych przeznaczonych specjalnie dla procedur zewnętrznych. Te właściwości nie są domyślnie dołączane, dlatego aby użyć którejkolwiek z nich, należy ręcznie zmodyfikować plik `agent.properties` . Jeśli podczas działania agenta zostanie wprowadzona zmiana w pliku `agent.properties` , zatrzymaj i zrestartuj agenta, aby zmiany zostały uwzględnione.

Procedury zewnętrzne monitora zasobów MFT

Procedury zewnętrzne monitora zasobów umożliwiają skonfigurowanie kodu niestandardowego, który ma być uruchamiany po spełnieniu warunku wyzwalacza monitora, zanim zostanie uruchomione powiązane zadanie.

Nie zaleca się wywoływania nowych operacji przesyłania bezpośrednio z kodu wyjścia użytkownika. W pewnych okolicznościach powoduje to wielokrotne przesyłanie plików, ponieważ programy zewnętrzne nie są odporne na restarty agenta.

Programy zewnętrzne monitora zasobów używają istniejącej infrastruktury dla programów zewnętrznych. Procedury zewnętrzne monitora są wywoływane po wyzwoleniu monitora, ale przed uruchomieniem odpowiedniego zadania przez zadanie monitora. Dzięki temu program użytkownika obsługi wyjścia może zmodyfikować zadanie, które ma zostać uruchomione, i zdecydować, czy zadanie ma być kontynuowane, czy nie. Zadanie monitorowania można zmodyfikować, aktualizując metadane monitora, które są następnie używane do podstawiania zmiennych w dokumencie zadania utworzonym w wyniku utworzenia oryginalnego monitora. Alternatywnie wyjście monitora może zastąpić lub zaktualizować łańcuch XML definicji zadania przekazany jako parametr. Wyjście monitora może zwrócić kod wyniku 'Kontynuuj' lub 'Anuluj' dla zadania. Jeśli zostanie zwrócona wartość Anuluj, zadanie nie zostanie uruchomione i monitor nie zostanie ponownie uruchomiony, dopóki monitorowany zasób nie spełni warunków wyzwalacza. Jeśli zasób nie uległ zmianie, wyzwalacz nie zostanie uruchomiony. Podobnie jak w przypadku innych programów zewnętrznych, można połączyć ze sobą programy zewnętrzne monitora. Jeśli jedno z wyjść zwraca kod wyniku anulowania, ogólnym wynikiem jest anulowanie, a zadanie nie jest uruchomione.

- Odwzorowanie metadanych środowiska (takie same jak w przypadku innych programów zewnętrznych)
- Odwzorowanie metadanych monitora, w tym niezmienialne metadane systemu i niezmienialne metadane użytkownika. Niezmienialne metadane systemu są następujące:
  - FILENAME-nazwa pliku spełniającego warunek wyzwalacza
  - FILEPATH-ścieżka do pliku spełniającego warunek wyzwalacza
  - FILESIZE (w bajtach-te metadane mogą nie być obecne)-wielkość pliku spełniającego warunek wyzwalacza
  - LASTMODIFIEDDATE (Local)-data ostatniej zmiany zbioru spełniającego warunek wyzwalacza. Data jest wyrażona jako data lokalna strefy czasowej, w jakiej działa agent, i ma format daty ISO 8601.
  - LASTMODIFIEDTIME (Local)-czas w formacie lokalnym ostatniej zmiany zbioru spełniającego warunek wyzwalacza. Godzina jest wyrażona w czasie miejscowym strefy czasowej, w jakiej działa agent, i ma format godziny ISO 8601.
  - LASTMODIFIEDDATEUTC-data w formacie uniwersalnym określająca, że zbiór spełniający warunek wyzwalacza został ostatnio zmieniony. Data jest wyrażona jako data lokalna przekształcona w datę UTC i ma format daty ISO 8601.
  - LASTMODIFIEDTIMEUTC-czas w formacie uniwersalnym, że plik, który spełniał warunek wyzwalacza został ostatnio zmieniony. Godzina jest wyrażona w czasie miejscowym przekształconym w czas UTC i ma format godziny ISO 8601.
  - AGENTNAME-nazwa agenta monitorowania
- Łańcuch XML reprezentujący zadanie, które ma zostać uruchomione w wyniku działania wyzwalacza monitora.

Wyjścia monitora zwracają następujące dane:

- Wskaźnik określający, czy należy kontynuować (kontynuować, czy anulować)
- Łańcuch, który ma zostać wstawiony do komunikatu dziennika spełniającego warunki wyzwalacza

W wyniku uruchomienia kodu wyjścia monitora metadane monitora i łańcuch XML definicji zadania, które zostały pierwotnie przekazane jako parametry, mogły również zostać zaktualizowane.

Wartość właściwości agenta monitorExitClasses (w pliku `agent.properties`) określa, które klasy wyjścia monitorowania mają zostać załadowane. Poszczególne klasy wyjścia są rozdzielane przecinkami. Na przykład:

```
monitorExitClasses=testExits.TestExit1,testExits.testExit2
```

Interfejs programu zewnętrznego monitora to:

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constant
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param monitorMetaData
     *            meta data to associate with the monitor. The meta data passed
     *            to this method can be altered, and the changes will be
     *            reflected in subsequent exit routine invocations. This map
     *            also contains keys with IBM reserved names.  These entries are
     *            defined in the <code>MonitorMetaDataConstants</code> class and
     *            have special semantics. The the values of the IBM reserved names
     *            cannot be modified by the exit
     *
     * @param taskDetails
     *            An XML String representing the task to be executed as a result of
     *            the monitor triggering. This XML string may be modified by the
     *            exit
     *
     * @return    a monitor exit result object which is used to determine if the
     *            task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                                Map<String, String> monitorMetaData,
                                Reference<String> taskDetails);
}
```

Stałe dla wartości zarezerwowanych przez IBMw metadanych monitora są następujące:

```
package com.ibm.wmqfte.exitroutine.api;

/**
* Constants for IBM reserved values placed into the monitor meta data
* maps used by the monitor exit routines.
*/
public interface MonitorMetaDataConstants {

/**
* The value associated with this key is the name of the trigger
* file associated with the monitor. Any modification performed
* to this property by user exit routines will be ignored.
*/
final String FILE_NAME_KEY = "FILENAME";

/**
* The value associated with this key is the path to the trigger
* file associated with the monitor. Any modification performed
* to this property by user exit routines will be ignored.
*/
final String FILE_PATH_KEY = "FILEPATH";

/**
* The value associated with this key is the size of the trigger
* file associated with the monitor. This will not be present in
* the cases where the size cannot be determined. Any modification
* performed to this property by user exit routines will be ignored.
```

```
*/
final String FILE_SIZE_KEY = "FILESIZE";

/**
* The value associated with this key is the local date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY = "LASTMODIFIEDDATE";

/**
* The value associated with this key is the local time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY = "LASTMODIFIEDTIME";

/**
* The value associated with this key is the UTC date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY_UTC = "LASTMODIFIEDDATEUTC";

/**
* The value associated with this key is the UTC time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY_UTC = "LASTMODIFIEDTIMEUTC";

/**
* The value associated with this key is the name of the agent on which
* the monitor is running. Any modification performed to this property by
* user exit routines will be ignored.
*/
final String MONITOR_AGENT_KEY = "AGENTNAME";

}
```

Przykładowa procedura zewnętrzna monitora

Ta przykładowa klasa implementuje interfejs MonitorExit . Ten przykład dodaje niestandardową zmienną podstawianą do metadanych monitora o nazwie *REDIRECTEDAGENT* , która zostanie zapełniona wartością LONDON , jeśli godzina dnia jest nieparzysta, i wartością PARIS dla godzin parzystych. Kod wyniku wyjścia monitora jest ustawiony tak, aby zawsze zwracał wartość `proceed`.

```
package com.ibm.wmqfte.monitor;

import java.util.Calendar;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.MonitorExit;
import com.ibm.wmqfte.exitroutine.api.MonitorExitResult;
import com.ibm.wmqfte.exitroutine.api.Reference;

/**
 * Example resource monitor user exit that changes the monitor mutable
 * metadata value between 'LONDON' and 'PARIS' depending on the hour of the day.
 *
 */
public class TestMonitorExit implements MonitorExit {

    // custom variable that will substitute destination agent
    final static String REDIRECTED_AGENT = "REDIRECTEDAGENT";

    public MonitorExitResult onMonitor(
Map<String, String> environmentMetaData,
            Map<String, String> monitorMetaData,
Reference<String> taskDetails) {

        // always succeed
        final MonitorExitResult result = MonitorExitResult.PROCEED_RESULT;
```

```
        final int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

        if (hour%2 == 1) {
            monitorMetaData.put(REDIRECTED_AGENT, "LONDON");
        } else {
            monitorMetaData.put(REDIRECTED_AGENT, "PARIS");
        }

        return result;
    }
}
```

Odpowiednie zadanie dla monitora, który korzysta ze zmiennej podstawianej *REDIRECTEDAGENT* , może wyglądać podobnie do poniższego:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>reportserver.com</hostName>
        <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1"
                 QMgr="QM1"/>
    <destinationAgent agent="${REDIRECTEDAGENT}"
                      QMgr="QM2"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="delete">
          <file>c:\sourcefiles\reports.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\reports.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Przed rozpoczęciem przesyłania wartość atrybutu `agent` elementu <destinationAgent> jest zastępowana wartością `LONDON` lub `PARIS`.

Zmienną podstawianą należy określić w klasie wyjścia monitora, a kod XML definicji zadania-wielkimi literami.

Pojęcia pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

“Metadane dla programów zewnętrznych MFT” na stronie 2190
Istnieją trzy różne typy metadanych, które można dostarczyć do procedur zewnętrznych dla produktu Managed File Transfer: środowisko, przesyłanie i metadane plików. Te metadane są prezentowane jako odwzorowania par klucz-wartość Java .

“Interfejsy Java dla programów zewnętrznych MFT” na stronie 2201
Tematy w tej sekcji zawierają informacje uzupełniające na temat interfejsów Java dla procedur zewnętrznych.

Źródłowe i docelowe procedury zewnętrzne MFT

Odsyłacze pokrewne

“MFT Właściwości agenta dla procedur zewnętrznych” na stronie 2199
Oprócz właściwości standardowych w pliku `agent.properties` istnieje kilka właściwości zaawansowanych przeznaczonych specjalnie dla procedur zewnętrznych. Te właściwości nie są domyślnie dołączane, dlatego aby użyć którejkolwiek z nich, należy ręcznie zmodyfikować plik `agent.properties` . Jeśli podczas działania agenta zostanie wprowadzona zmiana w pliku `agent.properties` , zatrzymaj i zrestartuj agenta, aby zmiany zostały uwzględnione.

MFT Właściwości agenta dla procedur zewnętrznych

Oprócz właściwości standardowych w pliku `agent.properties` istnieje kilka właściwości zaawansowanych przeznaczonych specjalnie dla procedur zewnętrznych. Te właściwości nie są domyślnie dołączane, dlatego aby użyć którejkolwiek z nich, należy ręcznie zmodyfikować plik `agent.properties` . Jeśli podczas działania agenta zostanie wprowadzona zmiana w pliku `agent.properties` , zatrzymaj i zrestartuj agenta, aby zmiany zostały uwzględnione.

Zmienne środowiskowe mogą być używane w niektórych właściwościach Managed File Transfer , które reprezentują położenia plików lub katalogów. Pozwala to na zmianę położenia plików lub katalogów używanych podczas uruchamiania części produktu, w zależności od zmian w środowisku, na przykład od tego, który użytkownik uruchamia proces. Więcej informacji na ten temat zawiera sekcja Zmienne środowiskowe we właściwościach produktu MFT.

Właściwości procedury zewnętrznej

Procedury użytkownika są wywoływane w kolejności podanej w poniższej tabeli. Więcej informacji na temat pliku `agent.properties` zawiera sekcja Zaawansowane właściwości agenta: Procedura użytkownika.

Tabela 885. Właściwości agenta dla programów zewnętrznych

Nazwa właściwości	**Opis**
sourceTransferEndExitKlasy	Określa rozdzielaną przecinkami listę klas, które implementują źródłową procedurę wyjściową zakończenia przesyłania.
sourceTransferStartExitKlasy	Określa rozdzielaną przecinkami listę klas, które implementują źródłową procedurę wyjściową rozpoczęcia przesyłania.
destinationTransferStartExitKlasy	Określa rozdzielaną przecinkami listę klas, które implementują docelową procedurę wyjściową użytkownika uruchamiającą przesyłanie.
destinationTransferEndExitKlasy	Określa rozdzielaną przecinkami listę klas, które implementują docelową procedurę zewnętrzną przesyłania.
Ścieżka exitClass	Określa specyficzną dla platformy, rozdzielaną znakami listę katalogów, które działają jako ścieżka klasy dla procedur zewnętrznych. Katalog wyjść agenta jest przeszukiwany przed wpisami w tej ścieżce klasy. Nawiasy, przecinki (,) i ukośniki odwrotne (\) są znakami specjalnymi w komendach MFT i muszą być poprzedzone znakiem ukośnika odwrotnego (\). Windows Ścieżki do plików w systemie Windows można określić przy użyciu podwójnych ukośników odwrotnych (\\) jako separatora lub przy użyciu pojedynczych ukośników (/). Na przykład: `exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar; C:/mycomp/mqft/exits/fileFilter.jar.` Wartość tej właściwości może zawierać zmienne środowiskowe.
exitNativeLibraryPath	Określa specyficzną dla platformy, rozdzielaną znakami listę katalogów, które działają jako rodzima ścieżka biblioteki dla procedur zewnętrznych. Wartość tej właściwości może zawierać zmienne środowiskowe.
monitorExitClasses	Umożliwia podanie rozdzielanej przecinkami listy klas implementujących procedurę zewnętrzną monitora. Więcej informacji zawiera temat Procedury zewnętrzne monitora zasobów produktu MFT.
protocolBridgeCredentialExitClasses	Umożliwia podanie rozdzielanej przecinkami listy klas implementujących procedurę zewnętrzną referencji mostu protokołu. Więcej informacji zawiera temat Odwzorowywanie referencji dla serwera plików za pomocą klas wyjścia.
protocolBridgePropertiesExitClasses	Umożliwia podanie rozdzielanej przecinkami listy klas implementujących procedurę zewnętrzną właściwości serwera mostu protokołu. Więcej informacji zawiera temat ProtocolBridgePropertiesExit2: Wyszukiwanie właściwości serwera plików protokołu.

Tabela 885. Właściwości agenta dla programów zewnętrznych (kontynuacja)	
Nazwa właściwości	**Opis**
IOExitClasses	Umożliwia podanie rozdzielanej przecinkami listy klas implementujących procedurę zewnętrzną we/wy. Należy podawać tylko klasy implementujące interfejs IOExit. Nie należy podawać klas implementujących inne interfejsy procedur zewnętrznych we/wy, na przykład IOExitResourcePath i IOExitChannel. Więcej informacji zawiera temat Korzystanie z procedur zewnętrznych we/wy przesyłania produktu MFT.

Kolejność wywoływania wyjścia

Wyjścia źródłowe i docelowe są wywoływane w następującej kolejności:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Łączenie wyjść źródłowych i docelowych w łańcuch

Jeśli określono wiele wyjść, najpierw wywoływane jest pierwsze wyjście z listy, następnie drugie wyjście itd. Wszystkie zmiany wprowadzone przez pierwsze wyjście są przekazywane jako dane wejściowe do wyjścia, które jest następnie wywoływane itd. Jeśli na przykład istnieją dwa źródłowe wyjścia początkowe przesyłania, wszystkie zmiany wprowadzone w metadanych przesyłania przez pierwsze wyjście są wprowadzane do drugiego wyjścia. Każde wyjście zwraca własny wynik. Jeśli wszystkie wyjścia danego typu zwrócą wartość PROCEED jako kod wyniku przesyłania, to cały wynik to PROCEED. Jeśli co najmniej jedno wyjście zwróci wartość CANCEL_TRANSFER, ogólnym wynikiem jest CANCEL_TRANSFER. Wszystkie kody wyników i łańcuchy zwracane przez wyjścia są zapisywane w dzienniku przesyłania.

Jeśli ogólnym wynikiem wyjścia rozpoczęcia przesyłania źródłowego jest PROCEED, przesyłanie jest kontynuowane przy użyciu zmian wprowadzonych przez wyjścia. Jeśli ogólnym wynikiem jest CANCEL_TRANSFER, wywoływane są źródłowe wyjścia końcowe przesyłania, a następnie przesyłanie jest anulowane. Status zakończenia w dzienniku przesyłania to "anulowany".

Jeśli ogólny wynik z docelowego wyjścia początkowego przesyłania jest kontynuowany, przesyłanie jest kontynuowane przy użyciu zmian wprowadzonych przez wyjścia. Jeśli ogólnym wynikiem jest CANCEL_TRANSFER, wywoływane są docelowe wyjścia końcowe przesyłania, a następnie wywoływane są źródłowe wyjścia końcowe przesyłania. Na koniec przesyłanie zostało anulowane. Status zakończenia w dzienniku przesyłania to "anulowany".

Jeśli wyjście źródłowe lub docelowe musi przekazać informacje do następujących wyjść w łańcuchu lub w kolejności wykonywania, należy to zrobić, aktualizując metadane przesyłania. Użycie metadanych przesyłania jest specyficzne dla implementacji wyjścia. Na przykład, jeśli wyjście ustawia wynik powrotu na CANCEL_TRANSFER i musi komunikować się z następującymi wyjściami, że operacja przesyłania została anulowana, należy to zrobić, ustawiając wartość metadanych przesyłania w sposób zrozumiały dla innych wyjść.

Przykład

```
sourceTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferStartExit
sourceTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferEndExit
destinationTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferStartExit
destinationTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferEndExit
exitClassPath=C:/mycomp/mqft/exits/encryptFileExit.jar;C:/mycomp/mqft/exits/fileFilter.jar
```

Pojęcia pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

“Metadane dla programów zewnętrznych MFT” na stronie 2190

Istnieją trzy różne typy metadanych, które można dostarczyć do procedur zewnętrznych dla produktu Managed File Transfer: środowisko, przesyłanie i metadane plików. Te metadane są prezentowane jako odwzorowania par klucz-wartość Java .

“Interfejsy Java dla programów zewnętrznych MFT” na stronie 2201
Tematy w tej sekcji zawierają informacje uzupełniające na temat interfejsów Java dla procedur zewnętrznych.

Odsyłacze pokrewne

“Procedury zewnętrzne monitora zasobów MFT” na stronie 2194
Procedury zewnętrzne monitora zasobów umożliwiają skonfigurowanie kodu niestandardowego, który ma być uruchamiany po spełnieniu warunku wyzwalacza monitora, zanim zostanie uruchomione powiązane zadanie.

Zmienne środowiskowe we właściwościach MFT

Plik MFT `agent.properties`

Interfejsy Java dla programów zewnętrznych MFT

Tematy w tej sekcji zawierają informacje uzupełniające na temat interfejsów Java dla procedur zewnętrznych.

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs IOExit.java” na stronie 2207

“Interfejs IOExitChannel.java” na stronie 2209

“Interfejs IOExitLock.java” na stronie 2210

“Interfejs IOExitPath.java” na stronie 2211

“Interfejs IOExitProperties.java” na stronie 2212

“Interfejs IOExitRecordChannel.java” na stronie 2215

“Interfejs IOExitRecordResourcePath.java” na stronie 2217

“Interfejs IOExitResourcePath.java” na stronie 2218

“Interfejs IOExitWildcardPath.java” na stronie 2222

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

“Interfejs ProtocolBridgeCredentialExit2.java” na stronie 2225

“Interfejs ProtocolBridgePropertiesExit2.java” na stronie 2226

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs SourceTransferEndExit.java” na stronie 2228

Interfejs CDCredentialExit.java

CDCredentialExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
```

```
 *   Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are invoked as part of
 * user exit routine processing.  This interface defines methods that are
 * invoked by a Connect:Direct bridge agent to map the IBM MQ user ID of the transfer to credentials
 * that are used to access the Connect:Direct node.
 * There will be one instance of each implementation class per Connect:Direct bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface CDCredentialExit {

    /**
     * Invoked once when a Connect:Direct bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *            The values of properties defined for the Connect:Direct bridge.
     *            These values can only be read, they cannot be updated by
     *            the implementation.
     *
     * @return    true if the initialisation is successful and false if unsuccessful
     *            If false is returned from an exit the Connect:Direct bridge agent does not
     *            start.
     */
     public boolean initialize(final Map<String, String> bridgeProperties);

     /**
      * Invoked once per transfer to map the IBM MQ user ID in the transfer message to the
      * credentials to be used to access the Connect:Direct node.
       *
       * @param mqUserId The IBM MQ user ID from which to map to the credentials to be used
      *                 to access the Connect:Direct node
     * @param snode    The name of the Connect:Direct SNODE specified as the cdNode in the
     *                 file path. This is used to map the correct user ID and password for the
      *                 SNODE.
     * @return         A credential exit result object that contains the result of the map and
      *                 the credentials to use to access the Connect:Direct node
      */

     public CDCredentialExitResult mapMQUserId(final String mqUserId, final String snode);

      /**
      * Invoked once when a Connect:Direct bridge agent is shutdown. This method releases
      * any resources that were allocated by the exit
      *
       * @param bridgeProperties
      *            The values of properties defined for the Connect:Direct bridge.
      *            These values can only be read, they cannot be updated by
       *            the implementation.
       *
       * @return
     */
     public void shutdown(final Map<String, String> bridgeProperties);      }
```

Interfejs CredentialExitResult.java

CredentialExitResult.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
```

```
 *   IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a Credential mapMQUserId exit method.  It is composed of a result
 * code, which determines whether the mapping of the user id was successful, and an optional
 * Credentials object if the mapping is successful.
 */
public class CredentialExitResult {

    private final CredentialExitResultCode resultCode;
    private final Credentials credentials;

    /**
     * Constructor.  Creates a credential exit result object with a specified result
     * code and optionally credentials.
     *
     * @param resultCode
     *            The result code to associate with the exit result being created.
     *
     * @param credentials
     *            The credentials to associate with the exit result being created.
     *            A value of <code>null</code> can be specified to indicate no
     *            credentials. If the resultCode is USER_SUCCESSFULLY_MAPPED the
     *            credentials must be set to a non-null value,
     */
    public CredentialExitResult(CredentialExitResultCode resultCode, Credentials credentials) {
        this.resultCode = resultCode;
        this.credentials = credentials;
    }


    /**
     * Returns the result code associated with this credential exit result
     *
     * @return    the result code associated with this exit result.
     */
    public CredentialExitResultCode getResultCode() {
        return resultCode;
    }

    /**
     * Returns the credentials associated with this credential exit result
     *
     * @return    the explanation associated with this credential exit result.
     */
    public Credentials getCredentials() {
        return credentials;
    }
}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs DestinationTransferEndExit.java

DestinationTransferEndExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
```

```
 *
 *   Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param transferExitResult
     *            a result object reflecting whether or not the transfer completed
     *            successfully.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.     This is the name of the agent that the
     *            implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The information can
     *            only be read, it cannot be updated by the implementation.  This
     *            map may also contain keys with IBM reserved names.  These
     *            entries are defined in the <code>TransferMetaDataConstants</code>
     *            class and have special semantics.
     *
     * @param fileResults
     *            a list of file transfer result objects that describe the source
     *            file name, destination file name and result of each file transfer
     *            operation attempted.
     *
     * @return    an optional description to enter into the log message describing
     *            transfer completion.  A value of <code>null</code> can be used
     *            when no description is required.
     */
    String onDestinationTransferEnd(TransferExitResult transferExitResult,
                    String sourceAgentName,
                    String destinationAgentName,
                    Map<String, String>environmentMetaData,
                    Map<String, String>transferMetaData,
                    List<FileTransferResult>fileResults);
}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs SourceTransferEndExit.java” na stronie 2228

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs DestinationTransferStartExit.java

DestinationTransferStartExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the destination of the transfer.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.     This is the name of the agent that the
     *            implementation of this method will be invoked from.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The information can
     *            only be read, it cannot be updated by the implementation.  This
     *            map may also contain keys with IBM reserved names.  These
     *            entries are defined in the <code>TransferMetaDataConstants</code>
     *            class and have special semantics.
     *
     * @param fileSpecs
     *            a list of file specifications that govern the file data to
     *            transfer.  The implementation of this method can modify the
     *            entries in this list and the changes will be reflected in the
     *            files transferred. However, new entries may not be added and
     *            existing entries may not be removed.
     *
     * @return    a transfer exit result object which is used to determine if the
     *            transfer should proceed, or be cancelled.
     */
    TransferExitResult onDestinationTransferStart(String sourceAgentName,
                                                  String destinationAgentName,
                                                Map<String, String> environmentMetaData,
                                                Map<String, String> transferMetaData,
                                                List<Reference<String>> fileSpecs);
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs SourceTransferEndExit.java” na stronie 2228

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs FileTransferResult.java

FileTransferResult.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * Result information about a file transfer.
 */
public interface FileTransferResult {

    /** An enumeration for the <code>getCorrelatorType()</code> method. */
    public enum CorrelationInformationType {
        /** No correlation information is available for this result */
        NONE,
        /**
         * The correlation information relates to work done in
         * IBM Sterling File Gateway.
         */
        SFG
    }

    /**
     * Returns the source file specification, from which the file was transferred.
     *
     * @return    the source file specification, from which the file was
     *            transferred.
     */
    String getSourceFileSpecification();

    /**
     * Returns the destination file specification, to which the file was transferred.
     *
     * @return    the destination file specification, to which the file was
     *            transferred.  A value of <code>null</code> may be returned
     *            if the transfer did not complete successfully.
     */
    String getDestinationFileSpecification();

    /**
     * Returns the result of the file transfer operation.
     *
     * @return    the result of the file transfer operation.
     */
    FileExitResult getExitResult();

    /**
     * @return an enumerated value that identifies the product to which this correlating
     *         information relates.
     */
    CorrelationInformationType getCorrelatorType();

    /**
     * @return the first string component of the correlating identifier that relates
     *         this transfer result to work done in another product.  A value of null
     *         may be returned either because the other product does not utilize a
     *         string based correlation information or because there is no correlation
     *         information.
     */
    String getString1Correlator();

    /**
```

```
     * @return the first long component of the correlating identifier that relates
     *         this transfer result to work done in another product.  A value of zero
     *         is returned when there is no correlation information or the other
     *         product does not utilize long based correlation information or because
     *         the value really is zero!
     */
    long getLong1Correlator();
}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs IOExit.java

IOExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

/**
 * An interface that is implemented by classes that you want to be invoked as
 * part of user exit routine processing. This interface defines methods that
 * will be invoked during transfers to perform the underlying file system I/O
 * work for WMQFTE transfers.
 * <p>
 * The {@link #initialize(Map)} method will be called once when the exit is
 * first installed. The WMQFTE agent properties are passed to this method, thus
 * enabling the exit to understand its environment.
 * <p>
 * The {@link #isSupported(String)} method will be invoked during WMQFTE
 * transfers to determine whether the user exit should be used. If the
 * {@link #isSupported(String)} method returns a value of {@code true}, the
 * {@link #newPath(String)} method will be invoked for the paths specified for
 * the transfer request. The returned {@link IOExitPath} instance from a
 * {@link #newPath(String)} method invocation will then be used by the WMQFTE
 * transfer to obtain information about the resource and to transfer data to or
 * from the resource.
 * <p>
 * To obtain transfer context for an I/O exit, a {@link SourceTransferStartExit}
 * or {@link DestinationTransferStartExit} as appropriate, should be installed
 * to enable information to be seen by this exit. The
 * {@link SourceTransferStartExit} or {@link DestinationTransferStartExit} are
 * passed the transfer's environment, metadata, and a list of file
 * specifications for the transfer. The paths for the file specifications are
 * the paths passed to the I/O exit's {@link #newPath(String)} method.
 * <p>
 * Note also that the {@link #isSupported(String)} and {@link #newPath(String)}
 * methods might be called at other times by a WMQFTE agent and not just during
 * transfers. For example, at transfer setup time the I/O system is queried to
```

```
 * resolve the full resource paths for transfer.
 */
public interface IOExit {

    /**
     * Invoked once when the I/O exit is first required for use. It is intended
     * to initialize any resources that are required by the exit.
     *
     * @param agentProperties
     *            The values of properties defined for the WMQFTE agent. These
     *            values can only be read, they cannot be updated by the
     *            implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an
     *         exit, the exit will not be used.
     */
    boolean initialize(final Map<String, String> agentProperties);

    /**
     * Indicates whether this I/O user exit supports the specified path.
     * <p>
     * This method is used by WMQFTE to determine whether the I/O user exit
     * should be used within a transfer. If no I/O user exit returns true for
     * this method, the default WMQFTE file I/O function will be used.
     *
     * @param path
     *            The path to the required I/O resource.
     * @return {@code true} if the specified path is supported by the I/O exit,
     *         {@code false} otherwise
     */
    boolean isSupported(String path);

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path.
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
     *            The path to the required I/O resource.
     * @return A {@link IOExitPath} instance for the specified path.
     * @throws IOException
     *             If the path cannot be created for any reason.
     */
    IOExitPath newPath(String path) throws IOException;

    /**
     * Obtains a new {@link IOExitPath} instance for the specified I/O resource
     * path and passes record format and length information required by the
     * WMQFTE transfer.
     * <p>
     * Typically this method will be called for the following cases:
     * <ul>
     * <li>A path where a call to {@link #newPath(String)} has previously
     * returned a {@link IOExitRecordResourcePath} instance and WMQFTE is
     * re-establishing a new {@link IOExitPath} instance for the path, from an
     * internally-serialized state. The passed recordFormat and recordLength
     * will be the same as those for the original
     * {@link IOExitRecordResourcePath} instance.</li>
     * <li>A transfer destination path where the source of the transfer is
     * record oriented. The passed recordFormat and recordLength will be the
     * same as those for the source.</li>
     * </ul>
     * The implementation can act on the record format and length information as
     * deemed appropriate. For example, for a destination agent if the
     * destination does not already exist and the source of the transfer is
     * record oriented, the passed recordFormat and recordLength information
     * could be used to create an appropriate record-oriented destination path.
     * If the destination path already exists, the passed recordFormat and
     * recordLength information could be used to perform a compatibility check
     * and throw an {@link IOException} if the path is not compatible. A
     * compatibility check could ensure that a record oriented path's record
     * format is the same as the passed record format or that the record length
     * is greater or equal to the passed record length.
     * <p>
     * This method will be invoked by WMQFTE only if the
     * {@link #isSupported(String)} method has been called for the path and
     * returned {@code true}.
     *
     * @param path
```

```
     *            The path to the required I/O resource.
     * @param recordFormat
     *            The advised record format.
     * @param recordLength
     *            The advised record length.
     * @return A {@link IOExitPath} instance for the specified path.
     * @throws IOException
     *             If the path cannot be created for any reason. For example,
     *             the passed record format or length is incompatible with the
     *             path's actual record format or length.
     */
    IOExitPath newPath(String path, RecordFormat recordFormat, int recordLength)
            throws IOException;
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitChannel.java

IOExitChannel.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables data to be read from or written to an
 * {@link IOExitResourcePath} resource.
 */
public interface IOExitChannel {

    /**
     * Obtains the data size for the associated {@link IOExitResourcePath} in
     * bytes.
     *
     * @return The data size in bytes.
     * @throws IOException
     *             If a problem occurs while attempting obtain the size.
     */
    long size() throws IOException;

    /**
     * Closes the channel, flushing any buffered write data to the resource and
     * releasing any locks.
     *
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while closing the resource.
     *             This means that WMQFTE can attempt to recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs. For example, the channel might
     *             already be closed.
     */
    void close() throws RecoverableIOException, IOException;

    /**
     * Reads data from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Data is copied into the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
```

```
     * number of bytes read.
     *
     * @param buffer
     *            The buffer that the data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while reading the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes data to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Data is copied from the buffer starting at its current position and up to
     * its limit. On return, the buffer's position is updated to reflect the
     * number of bytes written.
     *
     * @param buffer
     *            The buffer containing the data to be written.
     * @return The number of bytes written, which might be zero.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while writing the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Forces any updates to this channel's resource to be written to its
     * storage device.
     * <p>
     * This method is required to force changes to both the resource's content
     * and any associated metadata to be written to storage.
     *
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while performing the force.
     *             For a WMQFTE transfer this means that it will attempt to
     *             recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    void force() throws RecoverableIOException, IOException;

    /**
     * Attempts to lock the entire resource associated with the channel for
     * shared or exclusive access.
     * <p>
     * The intention is for this method not to block if the lock is currently
     * unavailable.
     *
     * @param shared
     *            {@code true} if a shared lock is required, {@code false} if an
     *            exclusive lock is required.
     * @return A {@link IOExitLock} instance representing the newly acquired
     *         lock or null if the lock cannot be obtained.
     * @throws IOException
     *             If a problem occurs while attempting to acquire the lock.
     */
    IOExitLock tryLock(boolean shared) throws IOException;
}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitLock.java

IOExitLock.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a lock on a resource for either shared or exclusive access.
 * {@link IOExitLock} instances are returned from
 * {@link IOExitChannel#tryLock(boolean)} calls and WMQFTE will request the
 * release of the lock at the appropriate time during a transfer. Additionally, when
 * a {@link IOExitChannel#close()} method is called it will be the
 * responsibility of the channel to release any associated locks.
 */
public interface IOExitLock {

    /**
     * Releases the lock.
     * <p>
     * After this method has been successfully called the lock is to be deemed as invalid.
     *
     * @throws IOException
     *             If the channel associated with the lock is not open or
     *             another problem occurs while attempting to release the lock.
     */
    void release() throws IOException;

    /**
     * Indicates whether this lock is valid.
     * <p>
     * A lock is considered valid until its @ {@link #release()} method is
     * called or the associated {@link IOExitChannel} is closed.
     *
     * @return {@code true} if this lock is valid, {@code false} otherwise.
     */
    boolean isValid();

    /**
     * @return {@code true} if this lock is for shared access, {@code false} if
     *         this lock is for exclusive access.
     */
    boolean isShared();
}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitPath.java

IOExitPath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
```

```
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an abstract path that can be inspected and queried by WMQFTE for
 * transfer purposes.
 * <p>
 * There are two types of path supported:
 * <ul>
 * <li>{@link IOExitResourcePath} - Represents a path that denotes a data
 * resource. For example, a file, directory, or group of database records.</li>
 * <li>{@link IOExitWildcardPath} - Represents a wildcard path that can be
 * expanded to multiple {@link IOExitResourcePath} instances.</li>
 * </ul>
 */
public abstract interface IOExitPath {

    /**
     * Obtains the abstract path as a {@link String}.
     *
     * @return The abstract path as a {@link String}.
     */
    String getPath();

    /**
     * Obtains the name portion of this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a name of {@code
     * file1.txt}.
     *
     * @return the name portion of this abstract path as a {@link String}.
     */
    String getName();

    /**
     * Obtains the parent path for this abstract path as a {@link String}.
     * <p>
     * For example, a UNIX-style file system implementation evaluates the
     * path {@code /home/fteuser/file1.txt} as having a parent path of {@code
     * /home/fteuser}.
     *
     * @return The parent portion of the path as a {@link String}.
     */
    String getParent();

    /**
     * Obtains the abstract paths that match this abstract path.
     * <p>
     * If this abstract path denotes a directory resource, a list of paths
     * for all resources within the directory are returned.
     * <p>
     * If this abstract path denotes a wildcard, a list of all paths
     * matching the wildcard are returned.
     * <p>
     * Otherwise null is returned, because this abstract path probably denotes a
     * single file resource.
     *
     * @return An array of {@IOExitResourcePath}s that
     *         match this path, or null if this method is not applicable.
     */
    IOExitResourcePath[] listPaths();
}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitProperties.java

IOExitProperties.java

```
/*
 *   Licensed Materials - Property of IBM
```

```
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Properties that determine how WMQFTE treats an {@link IOExitPath} for certain
 * aspects of I/O. For example, whether to use intermediate files.
 */
public class IOExitProperties {

    private boolean rereadSourceOnRestart = true;
    private boolean rechecksumSourceOnRestart = true;
    private boolean rechecksumDestinationOnRestart = true;
    private boolean useIntermediateFileAtDestination = true;
    private boolean requiresSingleThreadedChannelIO = false;

    /**
     * Determines whether the I/O exit implementation expects the resource to be
     * re-read from the start if a transfer is restarted.
     *
     * @return {@code true} if, on restart, the I/O exit expects the source
     *         resource to be opened at the beginning and re-read from the
     *         beginning (the {@link IOExitPath#openForRead(long)} method is
     *         always invoked with 0L as an argument). {@code false} if, on
     *         restart, the I/O exit expects the source to be opened at the
     *         offset that the source agent intends to start reading from (the
     *         {@link IOExitPath#openForRead(long)} method can be invoked with a
     *         non-zero value as its argument).
     */
    public boolean getRereadSourceOnRestart() {
        return rereadSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation expects
     * the resource to be re-read from the beginning if a transfer is restarted.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rereadSourceOnRestart
     *            {@code true} if, on restart, the I/O exit expects the source
     *            resource to be opened at the beginning and re-read from the
     *            beginning (the {@link IOExitPath#openForRead(long)} method
     *            is always invoked with 0L as an argument). {@code false}
     *            if, on restart, the I/O exit expects the source to be opened
     *            at the offset that the source agent intends to start reading
     *            from (the {@link IOExitPath#openForRead(long)} method can be
     *            invoked with a non-zero value as its argument).
     */
    public void setRereadSourceOnRestart(boolean rereadSourceOnRestart) {
        this.rereadSourceOnRestart = rereadSourceOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the source
     * resource to be re-checksummed if the transfer is restarted.
     * Re-checksumming takes place only if the
     * {@link #getRereadSourceOnRestart()} method returns {@code true}.
     *
     * @return {@code true} if, on restart, the I/O exit expects the already-
     *         transferred portion of the source to be re-checksummed for
     *         inconsistencies. Use this option in environments
     *         where the source could be changed during a restart. {@code
     *         false} if, on restart, the I/O exit does not require the
     *         already-transferred portion of the source to be re-checksummed.
     */
    public boolean getRechecksumSourceOnRestart() {
        return rechecksumSourceOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
```

```
     * the source resource to be re-checksummed if the transfer is restarted.
     * Re-checksumming takes place only if the
     * {@link #getRereadSourceOnRestart()} method returns {@code true}.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rechecksumSourceOnRestart
     *            {@code true} if, on restart, the I/O exit expects the already
     *            transferred portion of the source to be re-checksummed
     *            for inconsistencies. Use this option in environments
     *            where the source could be changed during a restart.
     *            {@code false} if, on restart, the I/O exit does not
     *            require the already-transferred portion of the source to be
     *            re-checksummed.
     */
    public void setRechecksumSourceOnRestart(boolean rechecksumSourceOnRestart) {
        this.rechecksumSourceOnRestart = rechecksumSourceOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the destination
     * resource to be re-checksummed if the transfer is restarted.
     *
     * @return {@code true} if, on restart, the I/O exit expects the already
     *         transferred portion of the destination to be re-checksummed to
     *         check for inconsistencies. This option should be used in
     *         environments where the destination could have been changed while
     *         a restart is occurring. {@code false} if, on restart, the I/O exit
     *         does not require the already transferred portion of the
     *         destination to be re-checksummed.
     */
    public boolean getRechecksumDestinationOnRestart() {
        return rechecksumDestinationOnRestart;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
     * the destination resource to be re-checksummed if the transfer is
     * restarted.
     * <p>
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param rechecksumDestinationOnRestart
     *            {@code true} if, on restart, the I/O exit expects the already-
     *            transferred portion of the destination to be re-checksummed
     *            for inconsistencies. Use this option in environments
     *            where the destination could have been changed during a
     *            restart. {@code false} if, on restart, the I/O exit does not
     *            require the already-transferred portion of the destination
     *            to be re-checksummed.
     */
    public void setRechecksumDestinationOnRestart(
            boolean rechecksumDestinationOnRestart) {
        this.rechecksumDestinationOnRestart = rechecksumDestinationOnRestart;
    }

    /**
     * Determines whether the I/O exit implementation requires the use of an
     * intermediate file when writing the data at the destination. The
     * intermediate file mechanism is typically used to prevent an incomplete
     * destination resource from being processed.
     *
     * @return {@code true} if data should be written to an intermediate file at
     *         the destination and then renamed (to the requested destination
     *         path name as specified in the transfer request) after the transfer is
     *         complete. {@code false} if data should be written directly to the
     *         requested destination path name without the use of an
     *         intermediate file.
     */
    public boolean getUseIntermediateFileAtDestination() {
        return useIntermediateFileAtDestination;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
     * the use of an intermediate file when writing the data at the destination.
     * The intermediate file mechanism is typically used to prevent an
     * incomplete destination resource from being processed.
     *
     * <p>
```

```
     * The default is {@code true}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param useIntermediateFileAtDestination
     *            {@code true} if data should be written to an intermediate file
     *            at the destination and then renamed (to the requested
     *            destination path name as specified in the transfer request) after
     *            the transfer is complete. {@code false} if data should be written
     *            directly to the requested destination path name without the
     *            use of an intermediate file
     */
    public void setUseIntermediateFileAtDestination(
            boolean useIntermediateFileAtDestination) {
        this.useIntermediateFileAtDestination = useIntermediateFileAtDestination;
    }

    /**
     * Determines whether the I/O exit implementation requires
     * {@link IOExitChannel} instances to be accessed by a single thread only.
     *
     * @return {@code true} if {@link IOExitChannel} instances are to be
     *         accessed by a single thread only.
     */
    public boolean requiresSingleThreadedChannelIO() {
        return requiresSingleThreadedChannelIO;
    }

    /**
     * Sets the value to determine whether the I/O exit implementation requires
     * channel operations for a particular instance to be accessed by a
     * single thread only.
     * <p>
     * For certain I/O implementations it is necessary that resource path
     * operations such as open, read, write, and close are invoked only from a
     * single execution {@link Thread}. When set {@code true}, WMQFTE ensures
     * that the following are invoked on a single thread:
     * <ul>
     * <li>{@link IOExitResourcePath#openForRead(long) method and all methods of
     * the returned {@link IOExitChannel} instance.</li>
     * <li>{@link IOExitResourcePath#openForWrite(boolean)) method and all
     * methods of the returned {@link IOExitChannel} instance.</li>
     * </ul>
     * <p>
     * This has a slight performance impact, hence enable single-threaded channel
     * I/O only when absolutely necessary.
     * <p>
     * The default is {@code false}. The I/O exit should call this method when
     * required to change this value.
     *
     * @param requiresSingleThreadedChannelIO
     *            {@code true} if {@link IOExitChannel} instances are to be
     *            accessed by a single thread only.
     */
    public void setRequiresSingleThreadedChannelIO(boolean requiresSingleThreadedChannelIO) {
        this.requiresSingleThreadedChannelIO = requiresSingleThreadedChannelIO;
    }
}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitRecordChannel.java

IOExitRecordChannel.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
```

```
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables records of data to be read from or written
 * to an {@link IOExitRecordResourcePath} resource.
 * <p>
 * This is an extension of the {@link IOExitChannel} interface such that the
 * {@link #read(java.nio.ByteBuffer)} and {@link #write(java.nio.ByteBuffer)}
 * methods are expected to deal in whole records of data only. That is, the
 * {@link java.nio.ByteBuffer} returned from the read method and passed to the
 * write method is assumed to contain one or more complete records.
 */
public interface IOExitRecordChannel extends IOExitChannel {

    /**
     * Reads records from this channel into the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data read.
     * <p>
     * Record data is copied into the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes read.
     * <p>
     * Only whole records are copied into the buffer.
     * <p>
     * For a fixed-record-format resource, this might be multiple records. The
     * amount of data in the return buffer does not necessarily need to be a
     * multiple of the record length, but the last record is still to be treated
     * as a complete record and padded as required by the caller.
     * <p>
     * For a variable-format resource, this is a single whole record of a size
     * corresponding to the amount of return data or multiple whole records with
     * all except the last being treated as records of maximum size.
     *
     * @param buffer
     *            The buffer that the record data is to be copied into.
     * @return The number of bytes read, which might be zero, or -1 if the end of
     *         data has been reached.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while reading the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs, for example, if the passed
     *             buffer is insufficient to contain at least one complete
     *             record). For a WMQFTE transfer this means that it will be
     *             failed.
     */
    int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

    /**
     * Writes records to this channel from the given buffer, starting at this
     * channel's current position, and updates the current position by the
     * amount of data written. The channel's resource is grown to accommodate
     * the data, if necessary.
     * <p>
     * Record data is copied from the buffer starting at its current position
     * and up to its limit. On return, the buffer's position is updated to
     * reflect the number of bytes written.
     * <p>
     * The buffer is expected to contain only whole records.
     * <p>
     * For a fixed-record-format resource, this might be multiple records and if
     * there is insufficient data in the buffer for a complete record, the
     * record is to be padded as required to complete the record.
     * <p>
     * For a variable-record format resource the buffer is normally expected to
     * contain a single record of length corresponding to the amount of data
     * within the buffer. However, if the amount of data within the buffer
     * exceeds the maximum record length, the implementation can either:
     * <ol>
     * <li>throw an {@link IOException} indicating that it cannot handle the
     * situation.</li>
     * <li>Consume a record's worth of data from the buffer, leaving the remaining
     * data within the buffer.</li>
     * <li>Consume all the buffer data and just write what it can to the current
     * record. This effectively truncates the data.</li>
```

```
     * <li>Consume all the buffer data and write to multiple records.</li>
     * </ol>
     *
     * @param buffer
     *            The buffer containing the data to be written.
     * @return The number of bytes written, which might be zero.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while writing the data. For a
     *             WMQFTE transfer this means that it will attempt to recover.
     * @throws IOException
     *             If some other I/O problem occurs. For a WMQFTE transfer this
     *             means that it will be failed.
     */
    int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

z/OS *Interfejs IOExitRecordResourcePath.java*

IOExitRecordResourcePath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a record-oriented data resource (for example,
 * a z/OS data set). It allows the data to be located, the record format to be
 * understood, and {@link IOExitRecordChannel} instances to be created for read
 * or write operations.
 */
public interface IOExitRecordResourcePath extends IOExitResourcePath {

    /**
     * Record formats for record-oriented resources.
     */
    public enum RecordFormat {
        FIXED, VARIABLE
    }

    /**
     * Obtains the record length for records that are maintained by the resource
     * denoted by this abstract path.
     * <p>
     * For a resource with fixed-length records, the data for each record read
     * and written is assumed to be this length.
     * <p>
     * For a resource with variable-length records, this is the maximum length
     * for a record's data.
     * <p>
     * This method should return a value greater than zero, otherwise it can
     * result in the failure of a WMQFTE transfer that involves this abstract
     * path.
     *
     * @return The record length, in bytes, for records maintained by the
     *         resource.
     */
    int getRecordLength();
```

```
    /**
     * Obtains record format, as a {@link RecordFormat} instance, for records
     * that are maintained by the resource denoted by this abstract path.
     *
     * @return A {@link RecordFormat} instance for the record format for records
     *         that are maintained by the resource denoted by this abstract
     *         path.
     */
    RecordFormat getRecordFormat();

    /**
     * Opens a {@link IOExitRecordChannel} instance for reading data from the
     * resource denoted by this abstract path. The current data byte position
     * for the resource is expected to be the passed position value, such that
     * when {@link IOExitRecordChannel#read(java.nio.ByteBuffer)} is called,
     * data starting from that position is read.
     * <p>
     * Note that the data byte read position will be on a record boundary.
     *
     * @param position
     *            The required data byte read position.
     * @return A new {@link IOExitRecordChannel} instance allowing data to be
     *         read from the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to open the
     *             resource for reading. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitRecordChannel openForRead(long position)
            throws RecoverableIOException, IOException;

    /**
     * Opens a {@link IOExitRecordChannel} instance for writing data to the
     * resource denoted by this abstract path. Writing of data, using the
     * {@link IOExitRecordChannel#write(java.nio.ByteBuffer)} method, starts at
     * either the beginning of the resource or end of the current data for the
     * resource, depending on the specified append parameter.
     *
     * @param append
     *            When {@code true} indicates that data written to the resource
     *            should be appended to the end of the current data. When
     *            {@code false} indicates that writing of data is to start at
     *            the beginning of the resource; any existing data is lost.
     * @return A new {@link IOExitRecordChannel} instance allowing data to be
     *         written to the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to open the
     *             resource for writing. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitRecordChannel openForWrite(boolean append)
            throws RecoverableIOException, IOException;
}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitResourcePath.java

IOExitResourcePath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   © Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
```

```
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a data resource (for example, a file,
 * directory, or group of database records). It allows the data to be located
 * and {@link IOExitChannel} instances to be created for read or write
 * operations.
 * <p>
 * There are two types of data resources as follows:
 * <ul>
 * <li>Directory - a container for other data resources. The
 * {@link #isDirectory()} method returns {@code true} for these.</li>
 * <li>File - a data container. This allows data to be read from or written to
 * it. The {@link #isFile()} method returns {@code true} for these.</li>
 * </ul>
 */
public interface IOExitResourcePath extends IOExitPath {

    /**
     * Creates a new {@link IOExitResourcePath} instance for a child path of the
     * resource denoted by this abstract path.
     * <p>
     * For example, with a UNIX-style path, {@code
     * IOExitResourcePath("/home/fteuser/test").newPath("subtest")} could be
     * equivalent to: {@code IOExitResourcePath("/home/fteuser/test/subtest")}
     *
     * @param child
     *            The child path name.
     * @return A new {@link IOExitResourcePath} instance that represents a child
     *         of this path.
     */
    IOExitResourcePath newPath(final String child);

    /**
     * Creates the directory path for the resource denoted by this abstract
     * path, including any necessary but nonexistent parent directories. If the
     * directory path already exists, this method has no effect.
     * <p>
     * If this operation fails, it might have succeeded in creating some of the
     * necessary parent directories.
     *
     * @throws IOException
     *             If the directory path cannot be fully created, when it does
     *             not already exist.
     */
    void makePath() throws IOException;

    /**
     * Obtains the canonical path of the abstract path as a {@link String}.
     * <p>
     * A canonical path is defined as being absolute and unique. For example,
     * the path can be represented as UNIX-style relative path: {@code
     * test/file.txt} but the absolute and unique canonical path representation
     * is: {@code /home/fteuser/test/file.txt}
     *
     * @return The canonical path as a {@link String}.
     * @throws IOException
     *             If the canonical path cannot be determined for any reason.
     */
    String getCanonicalPath() throws IOException;

    /**
     * Tests if this abstract path is an absolute path.
     * <p>
     * For example, a UNIX-style path, {@code /home/fteuser/test} is an absolute
     * path, whereas {@code fteuser/test} is not.
     *
     * @return {@code true} if this abstract path is an absolute path, {@code
     *         false} otherwise.
     */
    boolean isAbsolute();

    /**
     * Tests if the resource denoted by this abstract path exists.
     *
     * @return {@code true} if the resource denoted by this abstract path
     *         exists, {@code false} otherwise.
```

```
     * @throws IOException
     *             If the existence of the resource cannot be determined for any
     *             reason.
     */
    boolean exists() throws IOException;

    /**
     * Tests whether the calling application can read the resource denoted by
     * this abstract path.
     *
     * @return {@code true} if the resource for this path exists and can be
     *         read, {@code false} otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the
     *             resource can be read.
     */
    boolean canRead() throws IOException;

    /**
     * Tests whether the calling application can modify the resource denoted by
     * this abstract path.
     *
     * @return {@code true} if the resource for this path exists and can be
     *         modified, {@code false} otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the
     *             resource can be modified.
     */
    boolean canWrite() throws IOException;

    /**
     * Tests whether the specified user is permitted to read the resource
     * denoted by this abstract path.
     * <p>
     * When WMQFTE invokes this method, the user identifier is the MQMD user
     * identifier for the requesting transfer.
     *
     * @param userId
     *            User identifier to test for access.
     * @return {@code true} if the resource for this abstract path exists and is
     *         permitted to be read by the specified user, {@code false}
     *         otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the user
     *             is permitted to read the resource.
     */
    boolean readPermitted(String userId) throws IOException;

    /**
     * Tests whether the specified user is permitted to modify the resource
     * denoted by this abstract path.
     * <p>
     * When WMQFTE invokes this method, the user identifier is the MQMD user
     * identifier for the requesting transfer.
     *
     * @param userId
     *            User identifier to test for access.
     * @return {@code true} if the resource for this abstract path exists and is
     *         permitted to be modified by the specified user, {@code false}
     *         otherwise.
     * @throws IOException
     *             If a problem occurs while attempting to determine if the user
     *             is permitted to modify the resource.
     */
    boolean writePermitted(String userId) throws IOException;

    /**
     * Tests if the resource denoted by this abstract path is a directory-type
     * resource.
     *
     * @return {@code true} if the resource denoted by this abstract path is a
     *         directory type resource, {@code false} otherwise.
     */
    boolean isDirectory();

    /**
     * Creates the resource denoted by this abstract path, if it does not
     * already exist.
     *
     * @return {@code true} if the resource does not exist and was successfully
     *         created, {@code false} if the resource already existed.
     * @throws RecoverableIOException
```

```
     *             If a recoverable problem occurs while attempting to create
     *             the resource. This means that WMQFTE can attempt to recover
     *             the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    boolean createNewPath() throws RecoverableIOException, IOException;

    /**
     * Tests if the resource denoted by this abstract path is a file-type
     * resource.
     *
     * @return {@code true} if the resource denoted by this abstract path is a
     *         file type resource, {@code false} otherwise.
     */
    boolean isFile();

    /**
     * Obtains the last modified time for the resource denoted by this abstract
     * path.
     * <p>
     * This time is measured in milliseconds since the epoch (00:00:00 GMT,
     * January 1, 1970).
     *
     * @return The last modified time for the resource denoted by this abstract
     *         path, or a value of 0L if the resource does not exist or a
     *         problem occurs.
     */
    long lastModified();

    /**
     * Deletes the resource denoted by this abstract path.
     * <p>
     * If the resource is a directory, it must be empty for the delete to work.
     *
     * @throws IOException
     *             If the delete of the resource fails for any reason.
     */
    void delete() throws IOException;

    /**
     * Renames the resource denoted by this abstract path to the specified
     * destination abstract path.
     * <p>
     * The rename should still be successful if the resource for the specified
     * destination abstract path already exists and it is possible to replace
     * it.
     *
     * @param destination
     *            The new abstract path for the resource denoted by this
     *            abstract path.
     * @throws IOException
     *             If the rename of the resource fails for any reason.
     */
    void renameTo(IOExitResourcePath destination) throws IOException;

    /**
     * Creates a new path to use for writing to a temporary resource that did
     * not previously exist.
     * <p>
     * The implementation can choose the abstract path name for the temporary
     * resource. However, for clarity and problem diagnosis, the abstract path
     * name for the temporary resource should be based on this abstract path
     * name with the specified suffix appended and additional characters to make
     * the path unique (for example, sequence numbers), as required.
     * <p>
     * When WMQFTE transfers data to a destination it normally attempts to first
     * write to a temporary resource then on transfer completion renames the
     * temporary resource to the required destination. This method is called by
     * WMQFTE to create a new temporary resource path. The returned path should
     * be new and the resource should not previously exist.
     *
     * @param suffix
     *            Recommended suffix to use for the generated temporary path.
     *
     * @return A new {@link IOExitResourcePath} instance for the temporary
     *         resource path, that did not previously exist.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs whilst attempting to create
     *             the temporary resource. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
```

```
     *             If some other I/O problem occurs.
     */
    IOExitResourcePath createTempPath(String suffix)
            throws RecoverableIOException, IOException;

    /**
     * Opens a {@link IOExitChannel} instance for reading data from the resource
     * denoted by this abstract path. The current data byte position for the
     * resource is expected to be the passed position value, such that when
     * {@link IOExitChannel#read(java.nio.ByteBuffer)} is called, data starting
     * from that position is read.
     *
     * @param position
     *            The required data byte read position.
     * @return A new {@link IOExitChannel} instance allowing data to be read
     *         from the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs while attempting to open the
     *             resource for reading. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitChannel openForRead(long position) throws RecoverableIOException,
            IOException;

    /**
     * Opens a {@link IOExitChannel} instance for writing data to the resource
     * denoted by this abstract path. Writing of data, using the
     * {@link IOExitChannel#write(java.nio.ByteBuffer)} method, starts at either
     * the beginning of the resource or end of the current data for the
     * resource, depending on the specified append parameter.
     *
     * @param append
     *            When {@code true} indicates that data written to the resource
     *            should be appended to the end of the current data. When
     *            {@code false} indicates that writing of data is to start at
     *            the beginning of the resource; any existing data is lost.
     * @return A new {@link IOExitChannel} instance allowing data to be written
     *         to the resource denoted by this abstract path.
     * @throws RecoverableIOException
     *             If a recoverable problem occurs whilst attempting to open the
     *             resource for writing. This means that WMQFTE can attempt to
     *             recover the transfer.
     * @throws IOException
     *             If some other I/O problem occurs.
     */
    IOExitChannel openForWrite(boolean append) throws RecoverableIOException,
            IOException;

    /**
     * Tests if the resource denoted by this abstract path is in use by another
     * application. Typically, this is because another application has a lock on
     * the resource either for shared or exclusive access.
     *
     * @return {code true} if resource denoted by this abstract path is in use
     *         by another application, {@code false} otherwise.
     */
    boolean inUse();

    /**
     * Obtains a {@link IOExitProperties} instance for properties associated
     * with the resource denoted by this abstract path.
     * <p>
     * WMQFTE will read these properties to govern how a transfer behaves when
     * interacting with the resource.
     *
     * @return A {@link IOExitProperties} instance for properties associated
     *         with the resource denoted by this abstract path.
     */
    IOExitProperties getProperties();

}
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs IOExitWildcardPath.java

IOExitWildcardPath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents a path that denotes a wildcard. This can be used to match multiple
 * resource paths.
 */
public interface IOExitWildcardPath extends IOExitPath {
```

Zadania pokrewne

Korzystanie z procedur zewnętrznych we/wy przesyłania danych MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Interfejs MonitorExit.java

MonitorExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   Copyright IBM Corp. 2009, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

    /**
     * Invoked immediately prior to starting a task as the result of a monitor
     * trigger.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constant
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param monitorMetaData
     *            meta data to associate with the monitor. The meta data passed
     *            to this method can be altered, and the changes will be
     *            reflected in subsequent exit routine invocations. This map
     *            also contains keys with IBM reserved names.  These entries are
     *            defined in the <code>MonitorMetaDataConstants</code> class and
     *            have special semantics. The the values of the IBM reserved names
     *            cannot be modified by the exit
     *
```

```
     * @param taskDetails
     *            An XML String representing the task to be executed as a result of
     *            the monitor triggering. This XML string may be modified by the
     *            exit
     *
     * @return    a monitor exit result object which is used to determine if the
     *            task should proceed, or be cancelled.
     */
    MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                                Map<String, String> monitorMetaData,
                                Reference<String> taskDetails);
}
```

Zadania pokrewne

Monitorowanie zasobów MFT

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs SourceTransferEndExit.java” na stronie 2228

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs ProtocolBridgeCredentialExit.java

ProtocolBridgeCredentialExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing.  This interface defines methods that will
 * be invoked by a protocol bridge agent to map the MQ user ID of the transfer to credentials
 * that are to be used to access the protocol server.
 * There will be one instance of each implementation class per protocol bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to initialize
     * any resources that are required by the exit
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by
     *            the implementation.
     *
     *
     * @return    true if the initialization is successful and false if unsuccessful
     *            If false is returned from an exit the protocol bridge agent will not
     *            start
     */

   public boolean initialize(final Map<String>  bridgeProperties);
```

```
    /**
    * Invoked once for each transfer to map the MQ user ID in the transfer message to the
    * credentials to be used to access the protocol server
     *
     * @param mqUserId The MQ user ID from which to map to the credentials to be used
    *                 access the protocol server
    * @return         A credential exit result object that contains the result of the map and
    *                 the credentials to use to access the protocol server
    */

   public CredentialExitResult mapMQUserId(final String mqUserId);

   /**
      * Invoked once when a protocol bridge agent is shutdown. It is intended to release
      * any resources that were allocated by the exit
      *
       * @param bridgeProperties
      *            The values of properties defined for the protocol bridge.
      *            These values can only be read, they cannot be updated by
       *            the implementation.
     *
       * @return
   */

   public void shutdown(final Map<String> bridgeProperties);

}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odwzorowywanie referencji dla serwera plików przy użyciu klas wyjścia

Interfejs ProtocolBridgeCredentialExit2.java

ProtocolBridgeCredentialExit2.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * An interface that is implemented by classes that are invoked as part of user
 * exit routine processing. This interface defines methods that are invoked by a
 * protocol bridge agent to map the MQ user ID of the transfer to credentials
 * used to access a specified protocol bridge server. There will be one instance
 * of each implementation class for each protocol bridge agent. The methods can
 * be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit2 extends
        ProtocolBridgeCredentialExit {

    /**
     * Invoked once for each transfer to map the MQ user ID in the transfer
     * message to the credentials used to access a specified protocol server.
     *
     * @param endPoint
     *            Information that describes the protocol server to be accessed.
     * @param mqUserId
     *            The MQ user ID from which to map the credentials used to
     *            access the protocol server.
     * @return A {@link CredentialExitResult) instance that contains the result
     *         of the map and the credentials to use to access the protocol
     *         server.
     */
    public CredentialExitResult mapMQUserId(
```

```
                final ProtocolServerEndPoint endPoint, final String mqUserId);
 }
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odwzorowywanie referencji dla serwera plików przy użyciu klas wyjścia

Interfejs ProtocolBridgePropertiesExit2.java

ProtocolBridgePropertiesExit2.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2011, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit2 {

    /**
     * Invoked once when a protocol bridge agent is started. It is intended to
     * initialize any resources that are required by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     * @return {@code true} if the initialization is successful and {@code
     *         false} if unsuccessful. If {@code false} is returned from an exit
     *         the protocol bridge agent will not start.
     */
    public boolean initialize(final Map<String, String> bridgeProperties);

          /**
     * Invoked when the Protocol Bridge needs to access the protocol bridge credentials XML file.
     *
     * @return a {@link String} object giving the location of the ProtocolBridgeCredentials.xml
     */
    public String getCredentialLocation ();

    /**
     * Obtains a set of properties for the specified protocol server name.
     * <p>
     * The returned {@link Properties} must contain entries with key names
     * corresponding to the constants defined in
     * {@link ProtocolServerPropertyConstants} and in particular must include an
     * entry for all appropriate constants described as required.
     *
     * @param protocolServerName
     *            The name of the protocol server whose properties are to be
     *            returned. If a null or a blank value is specified, properties
     *            for the default protocol server are to be returned.
     * @return The {@link Properties} for the specified protocol server, or null
     *         if the server cannot be found.
```

```
     */
    public Properties getProtocolServerProperties(
            final String protocolServerName);

    /**
     * Invoked once when a protocol bridge agent is shut down. It is intended to
     * release any resources that were allocated by the exit.
     *
     * @param bridgeProperties
     *            The values of properties defined for the protocol bridge.
     *            These values can only be read, they cannot be updated by the
     *            implementation.
     */
    public void shutdown(final Map<String, String> bridgeProperties);

}
```

Zadania pokrewne

ProtocolBridgePropertiesExit: Wyszukiwanie właściwości serwera plików protokołu

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odwzorowywanie referencji dla serwera plików przy użyciu klas wyjścia

SourceFileExitFileklasaSpecification.java

SourceFileExitFileSpecification.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2012, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

    private final String sourceFileSpecification;
    private final String destinationFileSpecification;
    private final Map<String, String> sourceFileMetaData;
    private final Map<String, String> destinationFileMetaData;

    /**
     * Constructor.  Creates a source file exit file specification.
     *
     * @param sourceFileSpecification
     *            the source file specification to associate with the source file
     *            exit file specification.
     *
     * @param destinationFileSpecification
     *            the destination file specification to associate with the
     *            source file exit file specification.
     *
     * @param sourceFileMetaData
     *            the source file meta data.
     *
     * @param destinationFileMetaData
     *            the destination file meta data   .
     */
    public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                           final String destinationFileSpecification,
                                           final Map<String, String> sourceFileMetaData,
                                           final Map<String, String> destinationFileMetaData) {
        this.sourceFileSpecification = sourceFileSpecification;
```

```
        this.destinationFileSpecification = destinationFileSpecification;
        this.sourceFileMetaData = sourceFileMetaData;
        this.destinationFileMetaData = destinationFileMetaData;
    }

    /**
     * Returns the destination file specification.
     *
     * @return    the destination file specification.  This represents the location,
     *            on the agent acting as the destination for the transfer, where the
     *            file should be written.  Exit routines installed into the agent
     *            acting as the destination for the transfer may override this value.
     */
    public String getDestination() {
        return destinationFileSpecification;
    }

    /**
     * Returns the source file specification.
     *
     * @return    the source file specification.  This represents the location where
     *            the file data will be read from.
     */
    public String getSource() {
        return sourceFileSpecification;
    }

    /**
     * Returns the file meta data that relates to the source file specification.
     *
     * @return the file meta data that relates to the source file specification.
     */
    public Map<String, String> getSourceFileMetaData() {
        return sourceFileMetaData;
    }

    /**
     * Returns the file meta data that relates to the destination file specification.
     *
     * @return the file meta data that relates to the destination file specification.
     */
    public Map<String, String> getDestinationFileMetaData() {
        return destinationFileMetaData;
    }
}
```

Pojęcia pokrewne

“Metadane dla programów zewnętrznych MFT” na stronie 2190

Istnieją trzy różne typy metadanych, które można dostarczyć do procedur zewnętrznych dla produktu Managed File Transfer: środowisko, przesyłanie i metadane plików. Te metadane są prezentowane jako odwzorowania par klucz-wartość Java .

Interfejs SourceTransferEndExit.java

SourceTransferEndExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * source of the transfer.
```

```
 */
public interface SourceTransferEndExit {

    /**
     * Invoked immediately after the completion of a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param transferExitResult
     *            a result object reflecting whether or not the transfer completed
     *            successfully.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *            This is the name of the agent that the implementation of this
     *            method will be invoked from.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The information can
     *            only be read, it cannot be updated by the implementation.  This
     *            map may also contain keys with IBM reserved names.  These
     *            entries are defined in the <code>TransferMetaDataConstants</code>
     *            class and have special semantics.
     *
     * @param fileResults
     *            a list of file transfer result objects that describe the source
     *            file name, destination file name and result of each file transfer
     *            operation attempted.
     *
     * @return    an optional description to enter into the log message describing
     *            transfer completion.  A value of <code>null</code> can be used
     *            when no description is required.
     */
    String onSourceTransferEnd(TransferExitResult transferExitResult,
                    String sourceAgentName,
                    String destinationAgentName,
                    Map<String, String>environmentMetaData,
                    Map<String, String>transferMetaData,
                    List<FileTransferResult>fileResults);


}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“Interfejs SourceTransferStartExit.java” na stronie 2229

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs SourceTransferStartExit.java

SourceTransferStartExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
```

```
 *
 *   Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferStartExit {

    /**
     * Invoked immediately prior to starting a transfer on the agent acting as
     * the source of the transfer.
     *
     * @param sourceAgentName
     *            the name of the agent acting as the source of the transfer.
     *            This is the name of the agent that the implementation of this
     *            method will be invoked from.
     *
     * @param destinationAgentName
     *            the name of the agent acting as the destination of the
     *            transfer.
     *
     * @param environmentMetaData
     *            meta data about the environment in which the implementation
     *            of this method is running.  This information can only be read,
     *            it cannot be updated by the implementation.  The constants
     *            defined in <code>EnvironmentMetaDataConstants</code> class can
     *            be used to access the data held by this map.
     *
     * @param transferMetaData
     *            meta data to associate with the transfer.  The meta data passed
     *            to this method can be altered, and the changes to will be
     *            reflected in subsequent exit routine invocations.  This map may
     *            also contain keys with IBM reserved names.  These entries are
     *            defined in the <code>TransferMetaDataConstants</code> class and
     *            have special semantics.
     *
     * @param fileSpecs
     *            a list of file specifications that govern the file data to
     *            transfer.  The implementation of this method can add entries,
     *            remove entries, or modify entries in this list and the changes
     *            will be reflected in the files transferred.
     *
     * @return    a transfer exit result object which is used to determine if the
     *            transfer should proceed, or be cancelled.
     */
    TransferExitResult onSourceTransferStart(String sourceAgentName,
                          String destinationAgentName,
                          Map<String, String> environmentMetaData,
                          Map<String, String>transferMetaData,
                          List<SourceFileExitFileSpecification>fileSpecs);
}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

“SourceFileExitFileklasaSpecification.java” na stronie 2227

“Interfejs SourceTransferEndExit.java” na stronie 2228

“Interfejs DestinationTransferStartExit.java” na stronie 2204

“Interfejs DestinationTransferEndExit.java” na stronie 2203

“Interfejs MonitorExit.java” na stronie 2223

“Interfejs ProtocolBridgeCredentialExit.java” na stronie 2224

Interfejs TransferExitResult.java

TransferExitResult.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *    Copyright IBM Corp. 2008, 2024. All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */

package com.ibm.wmqfte.exitroutine.api;

/**
 * The result of invoking a transfer exit routine.  It is composed of a result
 * code, which determines if the transfer should proceed, and an optional explanatory
 * message.  The explanation, if present, is entered into the log message.
 */
public class TransferExitResult {

    private final TransferExitResultCode resultCode;
    private final String explanation;

    /**
     * For convenience, a static "proceed" result with no associated explanation
     * message.
     */
    public static final TransferExitResult PROCEED_RESULT =
        new TransferExitResult(TransferExitResultCode.PROCEED, null);

    /**
     * Constructor.  Creates a transfer exit result object with a specified result
     * code and explanation.
     *
     * @param resultCode
     *            The result code to associate with the exit result being created.
     *
     * @param explanation
     *            The explanation to associate with the exit result being created.
     *            A value of <code>null</code> can be specified to indicate no
     *            explanation.
     */
    public TransferExitResult(TransferExitResultCode resultCode, String explanation) {
        this.resultCode = resultCode;
        this.explanation = explanation;
    }

    /**
     * Returns the explanation associated with this transfer exit result.
     *
     * @return    the explanation associated with this exit result.
     */
    public String getExplanation() {
        return explanation;
    }

    /**
     * Returns the result code associated with this transfer exit result.
     *
     * @return    the result code associated with this exit result.
     */
    public TransferExitResultCode getResultCode() {
        return resultCode;
    }
}
```

Zadania pokrewne

Dostosowywanie programu MFT za pomocą programów zewnętrznych

Odsyłacze pokrewne

"Interfejs SourceTransferStartExit.java" na stronie 2229

"Interfejs DestinationTransferStartExit.java" na stronie 2204

"Interfejs DestinationTransferEndExit.java" na stronie 2203

"Interfejs MonitorExit.java" na stronie 2223

"Interfejs ProtocolBridgeCredentialExit.java" na stronie 2224

Formaty komunikatów, które można umieścić w kolejce komend agenta MFT

Te schematy XML definiują formaty komunikatów, które mogą być umieszczane w kolejce komend agenta w celu żądania wykonania działania przez agenta. Komunikat XML może zostać umieszczony w kolejce komend agenta za pomocą komend wiersza komend lub aplikacji.

- Format komunikatu żądania przesyłania plików
- Formaty komunikatów żądań monitora MFT
- Format komunikatu żądania agenta Ping MFT
- Format komunikatu odpowiedzi agenta MFT

Informacje dodatkowe dotyczące przesyłania komunikatów REST API

Informacje uzupełniające na temat messaging REST API.

Więcej informacji na temat korzystania z programu messaging REST APIzawiera sekcja Przesyłanie komunikatów przy użyciu programu REST API.

Zasoby REST API

Ta kolekcja tematów zawiera informacje uzupełniające o każdym z zasobów messaging REST API .

Więcej informacji na temat korzystania z programu messaging REST APIzawiera sekcja Przesyłanie komunikatów przy użyciu programu REST API.

/messaging/qmgr/{qmgrName}/queue/{queueName}/message

Interfejs REST API przesyłania komunikatów umożliwia umieszczanie komunikatów w kolejce lub przeglądanie ich lub destrukcyjne otrzymywanie z kolejki przy użyciu zasobu `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` .

POST /messaging/qmgr/{qmgrName}/queue/{queueName}/message

Aby umieścić komunikaty w określonej kolejce w określonym menedżerze kolejek, można użyć metody HTTP POST z zasobem `/messaging/qmgr/`*`{qmgrName}`*`/queue/`*`{queueName}`*`/message` .

Umieszcza komunikat IBM MQ zawierający treść żądania HTTP w określonym menedżerze kolejek i kolejce. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb. Ta metoda obsługuje tylko tekstowe treści żądań HTTP . Komunikaty są wysyłane jako komunikaty w formacie `MQSTR` lub `JMS TextMessage` i są umieszczane w bieżącym kontekście użytkownika.

V 9.3.0 Interfejs REST API V3 dodaje możliwość określenia zdefiniowanych przez użytkownika właściwości komunikatu i uwzględnienia priorytetu komunikatu. Nagłówki żądań ibm-mq-md-priority i ibm-mq-usr są dostępne tylko z interfejsem REST API V3. Nagłówek żądania ibm-mq-md-correlationId ma inny format w interfejsie REST API V3. Nagłówek może być identyfikatorem specyficznym dla aplikacji lub, jeśli zakodowany łańcuch wymaga przedrostka `ID:` . Jeśli żądanie POST zawiera komunikaty zdefiniowane przez użytkownika lub identyfikator korelacji specyficzny dla aplikacji, komunikat jest sformatowany jako `JMS TextMessage`.

Adres URL materiałów

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

V 9.3.0 `https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName
Określa nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie w celu przesyłania komunikatów. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb.

V 9.3.3 Z poziomu programu IBM MQ 9.3.3można nawiązać połączenie z lokalnym lub zdalnym menedżerem kolejek. Nazwa określona dla parametru **qmgrName** zależy od sposobu skonfigurowania serwera mqweb:

- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączenia tylko z lokalnymi menedżerami kolejek, należy użyć nazwy menedżera kolejek.
- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączeń ze zdalnymi menedżerami kolejek, należy użyć nazwy unikalnej dla menedżera kolejek zgodnie z informacjami o połączeniu ze zdalnym menedżerem kolejek.

Istnieje możliwość określenia, czy serwer mqweb jest skonfigurowany do nawiązywania połączeń z lokalnymi menedżerami kolejek, czy zdalnymi menedżerami kolejek, za pomocą komendy **dspmqweb properties** i wyświetlenia właściwości **mqRestMessagingConnectionMode** .

W nazwie menedżera kolejek jest rozróżniana wielkość liter.

Jeśli w nazwie menedżera kolejek znajdują się ukośniki, kropki lub znaki procentu, należy je zakodować w adresie URL:

- Ukośnik musi być zakodowany jako %2F.
- Kropka musi być zakodowana jako %2E.
- Znak procentu musi być zakodowany jako %25.

queueName
Określa nazwę kolejki, w której ma zostać umieszczony komunikat.

Kolejka musi być zdefiniowana jako lokalna, zdalna lub alias określonego menedżera kolejek-może również odwoływać się do kolejki klastrowej.

W nazwie kolejki rozróżniana jest wielkość liter.

Jeśli nazwa kolejki zawiera ukośnik lub znak procentu, znaki te muszą być zakodowane jako URL :

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

Jeśli włączone są połączenia HTTP , można użyć protokołu HTTP zamiast protokołu HTTPS . Więcej informacji na temat włączania protokołu HTTPzawiera sekcja Konfigurowanie portów HTTP i HTTPS.

Nagłówki żądań

Z żądaniem muszą być wysłane następujące nagłówki:

Autoryzacja
Ten nagłówek musi zostać wysłany, jeśli używane jest uwierzytelnianie podstawowe. Więcej informacji zawiera temat Korzystanie z podstawowego uwierzytelniania HTTP przy użyciu interfejsu REST API.

Content-Type
Ten nagłówek musi być wysłany z jedną z następujących wartości:

- `text/plain;charset=utf-8`
- `text/html;charset=utf-8`
- `text/xml;charset=utf-8`
- `application/json;charset=utf-8`
- `application/xml;charset=utf-8`

ibm-mq-rest-csrf-token
Ten nagłówek musi zostać wysłany. Może mieć dowolną wartość (nawet pustą).

Z żądaniem można opcjonalnie wysłać następujące nagłówki:

Accept-Language
Ten nagłówek określa wymagany język dla wszystkich wyjątków lub komunikatów o błędach zwracanych w treści komunikatu odpowiedzi.

REST API V2 REST API V1 **ibm-mq-md-correlationId**
Ten nagłówek ustawia identyfikator korelacji utworzonego komunikatu. Nagłówek jest określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty. Nie należy poprzedzać wartości przedrostkiem `"ID:"`, ponieważ interfejs REST API automatycznie dodaje ten łańcuch.

Na przykład:`ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02`

V 9.3.0 REST API V3 **ibm-mq-md-correlationId**
Ten nagłówek ustawia identyfikator korelacji utworzonego komunikatu. Identyfikator korelacji może mieć jedną z następujących postaci:

- 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty, poprzedzony łańcuchem `"ID:"`. Na przykład: `ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02`
- Wartość specyficzna dla aplikacji. Wartość jest łańcuchem specyficznym dla aplikacji: `ibm-mq-md-correlationId: My-Custom-CorrelId`

  Jeśli zostanie określona ta forma identyfikatora korelacji, miejscem docelowym komunikatu jest WMQ_CLIENT_JMS_COMPLIANT i dlatego zawiera on nagłówek MQRFH2 .

ibm-mq-md-utrata ważności
Ten nagłówek ustawia czas ważności dla utworzonego komunikatu. Utrata ważności komunikatu rozpoczyna się od momentu nadejścia komunikatu do kolejki. W wyniku tego opóźnienie sieci jest ignorowane. Nagłówek musi być określony jako jedna z następujących wartości:

Nieograniczony
Komunikat nie traci ważności.

Jest to wartość domyślna.

Liczba całkowita
Milisekundy przed utratą ważności komunikatu.

Ograniczone do zakresu od 0 do 99999999900.

ibm-mq-md-persistence
Ten nagłówek ustawia trwałość dla utworzonego komunikatu. Nagłówek jest określony jako jedna z następujących wartości:

nonPersistent
Komunikat nie przetrwa awarii systemu lub restartów menedżera kolejek.

Jest to wartość domyślna.

Trwały
Komunikat przetrwa awarie systemu lub restarty menedżera kolejek.

V 9.3.0 REST API V3 **ibm-mq-md-priorytet**

W przypadku interfejsu REST API V3ten nagłówek ustawia priorytet utworzonego komunikatu. Nagłówek jest określony jako jedna z następujących wartości:

asDestination
Komunikat używa priorytetu określonego w atrybucie DEFPRTY bazowego obiektu kolejki IBM MQ .

Jest to wartość domyślna.

Liczba całkowita
Określ rzeczywisty priorytet jako liczbę całkowitą z zakresu od 0 do 9.

Na przykład:`ibm-mq-md-priority: asDestination`

W przypadku interfejsu REST API V1 i interfejsu REST API V2priorytet komunikatu dla metody POST ma zawsze wartość 4.

ibm-mq-md-replyTo
Ten nagłówek ustawia miejsce docelowe odpowiedzi dla utworzonego komunikatu. Format nagłówka używa standardowej notacji dostarczania kolejki odpowiedzi i opcjonalnego menedżera kolejek: `replyQueue[@replyQmgr]`

Na przykład:`ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr`

V 9.3.0 REST API V3 **ibm-mq-usr**

Ten nagłówek ustawia zdefiniowane przez użytkownika właściwości komunikatu żądania.

Właściwości mają następującą składnię:

`ibm-mq-usr: property_name; user_value; user_type`

nazwa_właściwości
Nazwa określonej właściwości użytkownika. Musi to być poprawna nazwa właściwości JMS.

wartość_użytkownika
Wartość właściwości.

typ_użytkownika
Typ właściwości:

- `boolean` (prawda/fałsz, MQBOOL)
- `byte` (8-bitowa liczba całkowita, MQINT8)
- `short` (16-bitowa liczba całkowita, MQINT16)
- `integer` (32-bitowa liczba całkowita, MQINT32)
- `long` (64-bitowa liczba całkowita, MQINT64)
- `float` (32-bitowa liczba rzeczywista, MQFLOAT32)
- `double` (64-bitowa liczba rzeczywista, MQFLOAT64)
- `string` (łańcuch ujęty w cudzysłów)

W komunikacie można ustawić wiele właściwości. Można określić wiele właściwości rozdzielonych przecinkami w pojedynczym nagłówku żądania ibm-mq-usr lub użyć co najmniej dwóch oddzielnych instancji nagłówka żądania ibm-mq-usr.

Na przykład można ustawić wiele właściwości zdefiniowanych przez użytkownika w pojedynczym nagłówku: `ibm-mq-usr: userPropertyA;5;byte,userPropertyB;-10;integer`

Można również ustawić wiele właściwości zdefiniowanych przez użytkownika w oddzielnych instancjach nagłówka:`ibm-mq-usr: userPropertyA;5;byte ibm-mq-usr: userPropertyB;-10;integer`

Format treści żądania

Treść żądania musi być tekstem i musi być zakodowana w formacie UTF-8 . Nie jest wymagana żadna konkretna struktura tekstowa. Sformatowany komunikat MQSTR zawierający tekst treści żądania jest tworzony i umieszczany w określonej kolejce.

V 9.3.0 RESTAPI V3 Jeśli używane są zdefiniowane przez użytkownika właściwości interfejsu REST API w wersji V3 lub funkcje identyfikatora korelacji specyficzne dla aplikacji, tworzony jest sformatowany komunikat JMS `TextMessage` zawierający treść żądania i umieszczany w określonej kolejce.

Więcej informacji na ten temat zawierają przykłady.

Wymogi dotyczące bezpieczeństwa

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb. Role `MQWebAdmin` i `MQWebAdminRO` nie mają zastosowania do messaging REST API. Więcej informacji na temat zabezpieczeń dla produktu REST API zawiera sekcja Zabezpieczenia IBM MQ Console i REST API.

Po uwierzytelnieniu na serwerze mqweb użytkownik może używać zarówno produktu messaging REST API , jak i produktu administrative REST API.

Użytkownik zabezpieczeń programu wywołującego musi mieć możliwość umieszczania komunikatów w określonej kolejce:

- Kolejka określona przez część *{queueName}* adresu URL zasobu musi mieć włączoną opcję PUT.
- MQ Appliance ALW W przypadku kolejki określonej w części *{queueName}* zasobu URLnależy nadać uprawnienie +PUT użytkownikowi zabezpieczeń programu wywołującego.
- z/OS Dla kolejki określonej przez część *{queueName}* zasobu URLnależy nadać dostęp UPDATE użytkownikowi zabezpieczeń programu wywołującego.

Tą nazwą użytkownika zabezpieczeń może być użytkownik, który uruchomił serwer mqweb, lub użytkownik, który jest zalogowany na serwerze mqweb. Jeśli menedżer kolejek, z którym nawiązywane jest połączenie, jest zdalnym menedżerem kolejek, nazwą użytkownika zabezpieczeń może być użytkownik podany przez referencje w informacjach o połączeniu ze zdalnym menedżerem kolejek lub inny użytkownik określony przez reguły zabezpieczeń kanału. Więcej informacji na ten temat zawiera sekcja Określanie nazwy użytkownika używanej przez messaging REST API.

ALW W systemie AIX, Linux, and Windows można nadać uprawnienia użytkownikom zabezpieczeń, aby mogli korzystać z zasobów produktu IBM MQ, za pomocą komendy **setmqaut** . Więcej informacji zawiera temat **setmqaut** (nadawanie lub odbieranie uprawnienia).

z/OS W przypadku korzystania z systemu z/OS więcej informacji zawiera temat Konfigurowanie zabezpieczeń w systemie z/OS.

Jeśli z serwerem messaging REST APIużywane są zaawansowane zabezpieczenia komunikatów (Advanced Message Security-AMS), należy pamiętać, że wszystkie komunikaty są szyfrowane przy użyciu kontekstu serwera mqweb, a nie kontekstu użytkownika, który opublikuje komunikat.

Kody statusu odpowiedzi

201
Komunikat został pomyślnie utworzony i wysłany.

400
Podano niepoprawne dane.

Na przykład podano niepoprawną wartość nagłówka żądania.

401
Nie uwierzytelniono.

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb i musi być elementem co najmniej jednej z ról `MQWebAdmin`, `MQWebAdminRO` lub `MQWebUser`. Należy również określić nagłówek `ibm-mq-rest-csrf-token` . Aby uzyskać więcej informacji, zapoznaj się z sekcją: "Wymogi dotyczące bezpieczeństwa" na stronie 2236.

403
Brak uprawnień.

Program wywołujący został uwierzytelniony na serwerze mqweb i jest powiązany z poprawną nazwą użytkownika. Jednak użytkownik nie ma dostępu do wszystkich lub podzbioru wymaganych zasobów IBM MQ albo nie ma dostępu do roli `MQWebUser` . Więcej informacji na temat wymaganego dostępu zawiera sekcja "Wymogi dotyczące bezpieczeństwa" na stronie 2236.

404
Kolejka nie istnieje.

405
Kolejka jest zablokowana przez PUT.

415
Nagłówek lub treść komunikatu jest nieobsługiwanym typem nośnika.

Na przykład nagłówek `Content-Type` jest ustawiony na nieobsługiwany typ nośnika.

500
Problem z serwerem lub kod błędu z IBM MQ.

502
Bieżąca nazwa użytkownika zabezpieczeń nie może wysłać komunikatu, ponieważ dostawca przesyłania komunikatów nie obsługuje wymaganej funkcji. Na przykład, jeśli ścieżka klasy serwera mqweb jest niepoprawna.

503
Menedżer kolejek nie jest uruchomiony.

Nagłówki odpowiedzi

Z odpowiedzią są zwracane następujące nagłówki:

Treść-język
Określa identyfikator języka komunikatu odpowiedzi w przypadku wystąpienia błędów lub wyjątków. Używany w połączeniu z nagłówkiem żądania `Accept-Language` w celu wskazania języka wymaganego dla warunków błędu lub wyjątku. Jeśli żądany język nie jest obsługiwany, używana jest wartość domyślna serwera mqweb.

Content-Length (długość treści)
Określa długość treści odpowiedzi HTTP , nawet jeśli nie ma treści. Po pomyślnym zakończeniu wartość wynosi zero.

Content-Type
Określa typ treści odpowiedzi. Po pomyślnym zakończeniu wartością jest `text/plain;charset=utf-8`. W przypadku wystąpienia błędów lub wyjątków wartością jest `application/json;charset=utf-8`.

REST API V2 REST API V1 **ibm-mq-md-messageId**
Określa identyfikator komunikatu, który jest przydzielany przez program IBM MQ do tego komunikatu. Podobnie jak nagłówek żądania `ibm-mq-md-correlationId` , jest on reprezentowany jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
ibm-mq-md-messageId: 414d5120514d414445562020202020067d8ce5923582f07
```

V 9.3.0 REST API V3 **ibm-mq-md-messageId**

Określa identyfikator komunikatu, który jest przydzielany przez program IBM MQ do tego komunikatu. Podobnie jak nagłówek żądania `ibm-mq-md-correlationId` , jest on reprezentowany jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty i poprzedzony łańcuchem `ID:`.

Na przykład:

```
ibm-mq-md-messageId: ID:414d5120514d414445562020202020202067d8ce5923582f07
```

V 9.3.3 **ibm-mq-rozwiązany-qmgr**

Określa nazwę menedżera kolejek, który został użyty dla żądania. Jeśli w zasobie URLużyto nazwy unikalnej, nagłówek ten identyfikuje nazwę menedżera kolejek, która jest powiązana z tą nazwą unikalną. Jeśli unikalna nazwa używana w zasobie URL odwołuje się do grupy menedżerów kolejek, ten nagłówek identyfikuje, który menedżer kolejek w grupie był używany.

Format treści odpowiedzi

Treść odpowiedzi jest pusta, jeśli komunikat został wysłany pomyślnie. Jeśli wystąpi błąd, treść odpowiedzi zawiera komunikat o błędzie. Więcej informacji na ten temat zawiera sekcja Obsługa błędów w systemieREST API.

Przykłady

W poniższych przykładach stosowany jest adres URL zasobu w wersji 2. Jeśli używana jest wersja produktu IBM MQ wcześniejsza niż IBM MQ 9.1.5, należy zamiast tego użyć adresu URL zasobu v1. W adresie URL zasobu z przykładu należy wpisać v1 zamiast v2.

W poniższym przykładzie loguje się użytkownik o nazwie `mquser` z hasłem `mquser`. W przypadku komendy cURLżądanie logowania może wyglądać następująco: Windows . Znacznik LTPA jest zapisywany w pliku `cookiejar.txt` za pomocą opcji `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Po zalogowaniu się użytkownika znacznik LTPA i nagłówek `ibm-mq-rest-csrf-token` HTTP są używane do uwierzytelniania kolejnych żądań. `ibm-mq-rest-csrf-token` `token_value` może mieć dowolną wartość, w tym wartość pustą.

- Poniższy przykład komendy Windows cURL powoduje wysłanie komunikatu do kolejki `Q1` w menedżerze kolejek `QM1`przy użyciu opcji domyślnych. Komunikat zawiera tekst *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
  -H "Content-Type: text/plain;charset=utf-8" --data "Hello World!"
  ```

- W poniższym przykładzie komenda Windows cURL wysyła komunikat trwały do kolejki `Q1` w menedżerze kolejek `QM1`z utratą ważności 2 minut. Komunikat zawiera tekst *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
  -H "Content-Type: text/plain;charset=utf-8" -H "ibm-mq-md-persistence: persistent"
  -H "ibm-mq-md-expiry: 120000" --data "Hello World!"
  ```

- Poniższy przykład komendy Windows cURL powoduje wysłanie komunikatu nietrwałego do kolejki `Q1` w menedżerze kolejek `QM1`bez utraty ważności i ze zdefiniowanym identyfikatorem korelacji. Komunikat zawiera tekst *"Hello World!"*:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Content-Type: text/plain;charset=utf-8" -H "ibm-mq-md-persistence: nonPersistent"
  -H "ibm-mq-md-expiry: unlimited" -H "ibm-mq-md-correlationId:
  ```

```
414d5120514d414445562020202020202067d8b
f5923582e02" --data "Hello World!"
```

GET /messaging/qmgr/{qmgrName}/queue/{queueName}/message

Do przeglądania komunikatów z powiązanego menedżera kolejek i kolejki można użyć metody GET HTTP z zasobem `/messaging/qmgr/{qmgrName}/queue/{queueName}/message` .

Przegląda pierwszy dostępny komunikat z określonego menedżera kolejek i kolejki. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb. Treść komunikatu jest zwracana w treści odpowiedzi HTTP . Komunikat musi mieć format `MQSTR` lub `JMS TextMessage` i jest odbierany przy użyciu bieżącego kontekstu użytkownika.

Wszystkie komunikaty są pozostawiane w kolejce i do programu wywołującego zwracany jest odpowiedni kod statusu dla nieodpowiednich komunikatów. Na przykład komunikat, który nie ma formatu `MQSTR` lub `JMS TextMessage` .

V 9.3.0 Interfejs REST API V3 dodaje możliwość określania zdefiniowanych przez użytkownika właściwości komunikatu i dołączania priorytetu komunikatu do komunikatów. Nagłówki odpowiedzi ibm-mq-md-priority i ibm-mq-usr są dostępne tylko z interfejsem REST API V3. Nagłówek żądania ibm-mq-md-correlationId ma inny format w interfejsie REST API V3. Nagłówek może być identyfikatorem specyficznym dla aplikacji lub, jeśli jest to zakodowany łańcuch, zachowuje przedrostek `ID:` . Nagłówek odpowiedzi i parametr zapytania ibm-mq-md-messageId mają inny format w interfejsie REST API V3, zachowuje przedrostek `ID:` .

Adres URL materiałów

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

V 9.3.0 `https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/message`

qmgrName

Określa nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie w celu przesyłania komunikatów. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb.

V 9.3.3 Z poziomu programu IBM MQ 9.3.3można nawiązać połączenie z lokalnym lub zdalnym menedżerem kolejek. Nazwa określona dla parametru **qmgrName** zależy od sposobu skonfigurowania serwera mqweb:

- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączenia tylko z lokalnymi menedżerami kolejek, należy użyć nazwy menedżera kolejek.
- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączeń ze zdalnymi menedżerami kolejek, należy użyć nazwy unikalnej dla menedżera kolejek zgodnie z informacjami o połączeniu ze zdalnym menedżerem kolejek.

Istnieje możliwość określenia, czy serwer mqweb jest skonfigurowany do nawiązywania połączeń z lokalnymi menedżerami kolejek, czy zdalnymi menedżerami kolejek, za pomocą komendy **dspmqweb properties** i wyświetlenia właściwości **mqRestMessagingConnectionMode** .

W nazwie menedżera kolejek jest rozróżniana wielkość liter.

Jeśli w nazwie menedżera kolejek znajdują się ukośniki, kropki lub znaki procentu, należy je zakodować w adresie URL:

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

queueName

Określa nazwę kolejki, z której ma być przeglądany komunikat.

Kolejka musi być zdefiniowana jako lokalna lub jako alias wskazujący kolejkę lokalną.

W nazwie kolejki rozróżniana jest wielkość liter.

Jeśli nazwa kolejki zawiera ukośnik lub znak procentu, znaki te muszą być zakodowane jako URL :

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

Jeśli włączone są połączenia HTTP , można użyć protokołu HTTP zamiast protokołu HTTPS . Więcej informacji na temat włączania protokołu HTTPzawiera sekcja Konfigurowanie portów HTTP i HTTPS.

Opcjonalne parametry zapytania

REST API V2 REST API V1 **correlationId=*hexValue***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem korelacji.

hexValue

Parametr zapytania musi być określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

V 9.3.0 REST API V3 **correlationId= ID:*hexValue* lub correlationId=*wartość_specyfikacji_aplikacji***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem korelacji.

hexValue

Parametr zapytania musi być określony jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty i poprzedzony łańcuchem `"ID:"`.

Na przykład:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

wartość_specyfikacji_aplikacji

Parametr zapytania można określić jako łańcuch specyficzny dla aplikacji.

Na przykład:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=*hexValue***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem komunikatu.

hexValue
Parametr zapytania musi być określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

V 9.3.0 REST API V3 **messageId= ID:*hexValue***
Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem komunikatu.

hexValue
Parametr zapytania musi być określony jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty i poprzedzony łańcuchem `"ID:"`.

Na przykład:

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

Nagłówki żądań

Z żądaniem muszą być wysłane następujące nagłówki:

Autoryzacja
Ten nagłówek musi zostać wysłany, jeśli używane jest uwierzytelnianie podstawowe. Więcej informacji zawiera temat Korzystanie z podstawowego uwierzytelniania HTTP przy użyciu interfejsu REST API.

ibm-mq-rest-csrf-token
Ten nagłówek musi zostać wysłany. Może mieć dowolną wartość (nawet pustą).

Z żądaniem można opcjonalnie wysłać następujące nagłówki:

Akceptuj-zestaw znaków
Ten nagłówek może być używany do wskazywania, który zestaw znaków jest akceptowalny dla odpowiedzi. Jeśli ten nagłówek jest określony, musi być ustawiony jako `UTF-8`.

Accept-Language
Ten nagłówek określa wymagany język dla wszystkich wyjątków lub komunikatów o błędach zwracanych w treści komunikatu odpowiedzi.

Format treści żądania

Brak.

Wymogi dotyczące bezpieczeństwa

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb. Role `MQWebAdmin` i `MQWebAdminRO` nie mają zastosowania do messaging REST API. Więcej informacji na temat zabezpieczeń dla produktu REST API zawiera sekcja Zabezpieczenia IBM MQ Console i REST API.

Po uwierzytelnieniu na serwerze mqweb użytkownik może używać zarówno produktu messaging REST API , jak i produktu administrative REST API.

Nazwa użytkownika zabezpieczeń osoby nawiązującej połączenie musi mieć przypisane uprawnienie umożliwiające przeglądanie komunikatów z określonej kolejki:

- Kolejka określona przez część *{queueName}* adresu URL zasobu musi mieć włączoną opcję `BROWSE`.
- ALW MQ Appliance W przypadku kolejki określonej przez część *{queueName}* adresu URL zasobu należy nadać uprawnienia `+GET`, `+INQ` i `+BROWSE` jednostce głównej zabezpieczeń programu wywołującego.

- z/OS W przypadku kolejki określonej przez część *{queueName}* adresu URL zasobu, UPDATE, należy zezwolić na dostęp jednostce głównej zabezpieczeń programu wywołującego.

Tą nazwą użytkownika zabezpieczeń może być użytkownik, który uruchomił serwer mqweb, lub użytkownik, który jest zalogowany na serwerze mqweb. Jeśli menedżer kolejek, z którym nawiązywane jest połączenie, jest zdalnym menedżerem kolejek, nazwą użytkownika zabezpieczeń może być użytkownik podany przez referencje w informacjach o połączeniu ze zdalnym menedżerem kolejek lub inny użytkownik określony przez reguły zabezpieczeń kanału. Więcej informacji na ten temat zawiera sekcja Określanie nazwy użytkownika używanej przez messaging REST API.

ALW W systemie AIX, Linux, and Windows można nadać uprawnienia użytkownikom zabezpieczeń, aby mogli korzystać z zasobów produktu IBM MQ, za pomocą komendy **setmqaut** . Więcej informacji zawiera temat **setmqaut** (nadawanie lub odbieranie uprawnienia).

z/OS W przypadku korzystania z systemu z/OS więcej informacji zawiera temat Konfigurowanie zabezpieczeń w systemie z/OS.

Kody statusu odpowiedzi

200
Komunikat został odebrany pomyślnie.

204
Brak komunikatu.

400
Podano niepoprawne dane.

Na przykład podano niepoprawną wartość parametru zapytania.

401
Nie uwierzytelniono.

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb i musi być elementem co najmniej jednej z ról `MQWebAdmin`, `MQWebAdminRO` lub `MQWebUser`. Należy również określić nagłówek `ibm-mq-rest-csrf-token` . Aby uzyskać więcej informacji, zapoznaj się z sekcją: “Wymogi dotyczące bezpieczeństwa” na stronie 2241.

403
Brak uprawnień.

Program wywołujący został uwierzytelniony na serwerze mqweb i jest powiązany z poprawną nazwą użytkownika. Jednak użytkownik nie ma dostępu do wszystkich lub podzbioru wymaganych zasobów IBM MQ albo nie ma dostępu do roli `MQWebUser` . Więcej informacji na temat wymaganego dostępu zawiera sekcja “Wymogi dotyczące bezpieczeństwa” na stronie 2241.

404
Kolejka nie istnieje.

500
Problem z serwerem lub kod błędu z IBM MQ.

501
Nie można utworzyć odpowiedzi HTTP .

Na przykład odebrany komunikat ma niepoprawny typ lub ma poprawny typ, ale nie można przetworzyć treści.

502
Bieżąca nazwa użytkownika zabezpieczeń nie może odebrać komunikatu, ponieważ dostawca przesyłania komunikatów nie obsługuje wymaganej funkcji. Na przykład, jeśli ścieżka klasy serwera mqweb jest niepoprawna.

503
Menedżer kolejek nie jest uruchomiony.

Nagłówki odpowiedzi

Z odpowiedzią są zwracane następujące nagłówki:

Treść-język
Określa identyfikator języka komunikatu odpowiedzi w przypadku wystąpienia błędów lub wyjątków. Używany w połączeniu z nagłówkiem żądania `Accept-Language` w celu wskazania języka wymaganego dla warunków błędu lub wyjątku. Jeśli żądany język nie jest obsługiwany, używana jest wartość domyślna serwera mqweb.

Content-Length (długość treści)
Określa długość treści odpowiedzi HTTP , nawet jeśli nie ma treści. Wartość zawiera długość (w bajtach) danych komunikatu.

Content-Type
Określa typ treści zwracanej w treści odpowiedzi odebranego komunikatu. Po pomyślnym zakończeniu wartością jest `text/plain;charset=utf-8`. W przypadku wystąpienia błędów lub wyjątków wartością jest `application/json;charset=utf-8`.

REST API V2 REST API V1 **ibm-mq-md-correlationId**
Określa identyfikator korelacji odebranego komunikatu. Nagłówek jest zwracany, jeśli odebrany komunikat zawiera poprawny identyfikator korelacji. Jest on reprezentowany jako 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty.

Na przykład:

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

V 9.3.0 REST API V3 **ibm-mq-md-correlationId**
Określa identyfikator korelacji odebranego komunikatu. Nagłówek jest zwracany, jeśli odebrany komunikat zawiera poprawny identyfikator korelacji. Identyfikator korelacji może mieć jedną z następujących postaci:

- 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty, poprzedzony łańcuchem `"ID:"`. Na przykład:

  ```
  ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
  ```

- Wartość specyficzna dla aplikacji. Wartość jest łańcuchem specyficznym dla aplikacji:

  ```
  ibm-mq-md-correlationId: My-Custom-CorrelId
  ```

ibm-mq-md-utrata ważności
Określa pozostały czas ważności odebranego komunikatu. Nagłówek może mieć jedną z następujących wartości:

Nieograniczony
Komunikat nie traci ważności.

Liczba całkowita
Pozostała liczba milisekund przed utratą ważności komunikatu.

REST API V2 REST API V1 **ibm-mq-md-messageId**
Określa identyfikator komunikatu, który jest przydzielany przez program IBM MQ do tego komunikatu. Podobnie jak nagłówek `ibm-mq-md-correlationId` , jest on reprezentowany jako 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty.

Na przykład:

```
ibm-mq-md-messageId: 414d5120514d4144455620202020202067d8ce5923582f07
```

V 9.3.0 RESTAPI V3 **ibm-mq-md-messageId**

Określa identyfikator komunikatu, który jest przydzielany przez program IBM MQ do tego komunikatu. Podobnie jak nagłówek `ibm-mq-md-correlationId` , jest on reprezentowany jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty, z przedrostkiem `"ID:"`

Na przykład:

```
ibm-mq-md-messageId: ID:414d5120514d414445562020202020202067d8ce5923582f07
```

ibm-mq-md-persistence

Określa trwałość odebranego komunikatu. Nagłówek może mieć jedną z następujących wartości:

nonPersistent
Komunikat nie przetrwa awarii systemu lub restartów menedżera kolejek.

Trwały
Komunikat przetrwa awarie systemu lub restarty menedżera kolejek.

V 9.3.0 RESTAPI V3 **ibm-mq-md-priorytet**

Zwraca ustawienie priorytetu komunikatu. Na przykład:

```
ibm-mq-md-priority: 3
```

ibm-mq-md-replyTo

Określa miejsce docelowe odpowiedzi dla odebranego komunikatu. W formacie nagłówka używana jest standardowa notacja kolejki odpowiedzi i menedżera kolejek `replyQueue@replyQmgr`.

Na przykład:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

V 9.3.3 **ibm-mq-rozwiązany-qmgr**

Określa nazwę menedżera kolejek, który został użyty dla żądania. Jeśli w zasobie URLużyto nazwy unikalnej, nagłówek ten identyfikuje nazwę menedżera kolejek, która jest powiązana z tą nazwą unikalną. Jeśli unikalna nazwa używana w zasobie URL odwołuje się do grupy menedżerów kolejek, ten nagłówek identyfikuje, który menedżer kolejek w grupie był używany.

V 9.3.0 RESTAPI V3 **ibm-mq-usr**

Zwraca właściwości komunikatu zdefiniowane przez użytkownika. W komunikacie można ustawić wiele właściwości. W takim przypadku będą dwie lub więcej oddzielnych instancji nagłówka odpowiedzi ibm-mq-usr.

Na przykład:

```
ibm-mq-usr: myIProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp;true;boolean
```

Właściwości mają następującą składnię:

```
ibm-mq-usr: property_name; user_value; user_type
```

nazwa_właściwości
Nazwa określonej właściwości użytkownika. Musi to być poprawna nazwa właściwości JMS.

wartość_użytkownika
Wartość właściwości.

typ_użytkownika
Typ właściwości:

- `boolean` (prawda/fałsz, MQBOOL)
- `byte` (8-bitowa liczba całkowita, MQINT8)

- `short` (16-bitowa liczba całkowita, MQINT16)
- `integer` (32-bitowa liczba całkowita, MQINT32)
- `long` (64-bitowa liczba całkowita, MQINT64)
- `float` (32-bitowa liczba rzeczywista, MQFLOAT32)
- `double` (64-bitowa liczba rzeczywista, MQFLOAT64)
- `string` (łańcuch ujęty w cudzysłów)

Format treści odpowiedzi

Po pomyślnym zakończeniu treść odpowiedzi zawiera treść odebranego komunikatu. Jeśli wystąpi błąd, treść odpowiedzi zawiera sformatowany komunikat o błędzie JSON. Obie odpowiedzi są zakodowane jako UTF-8 . Więcej informacji na ten temat zawiera sekcja Obsługa błędów w systemieREST API.

Należy pamiętać, że podczas odbierania komunikatu obsługiwane są tylko komunikaty w formacie IBM MQ `MQSTR` lub `JMS TextMessage` .

Przeglądanie kolejki, która została oznaczona jako zablokowana za pomocą komendy GET, nie zwraca treści.

Jeśli przeglądana kolejka zawiera komunikaty ze zduplikowanymi identyfikatorami, podczas filtrowania według identyfikatora komunikatu zwracany jest pierwszy komunikat.

Przykłady

W poniższych przykładach stosowany jest adres URL zasobu w wersji 2. Jeśli używana jest wersja produktu IBM MQ wcześniejsza niż IBM MQ 9.1.5, należy zamiast tego użyć adresu URL zasobu v1. W adresie URL zasobu z przykładu należy wpisać v1 zamiast v2.

W poniższym przykładzie loguje się użytkownik o nazwie `mquser` z hasłem `mquser`. W przypadku komendy cURLżądanie logowania może wyglądać następująco: Windows . Znacznik LTPA jest zapisywany w pliku `cookiejar.txt` za pomocą opcji `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Po zalogowaniu się użytkownika znacznik LTPA i nagłówek `ibm-mq-rest-csrf-token` HTTP są używane do uwierzytelniania kolejnych żądań. `ibm-mq-rest-csrf-token` `token_value` może mieć dowolną wartość, w tym wartość pustą.

- Poniższy przykład komendy Windows cURL powoduje przeglądanie następnego dostępnego komunikatu z kolejki `Q1` w menedżerze kolejek `QM1`przy użyciu opcji domyślnych:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- REST API V2 REST API V1 Poniższy przykład komendy Windows cURL powoduje wyświetlenie komunikatu o konkretnym identyfikatorze korelacji `00000000000000000000000000000000000000000000abcdabcdz` kolejki `Q1` w menedżerze kolejek `QM1`:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
  correlationId=00000000000000000000000000000000000000000000abcdabcd"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: text/plain"
  ```

- V 9.3.0 REST API V3

  Poniższy przykład komendy Windows cURL jest taki sam jak w poprzednim przykładzie, ale używa interfejsu REST API V3. W przykładzie wyświetlane są tylko te komunikaty, które mają odpowiedni

identyfikator korelacji, 000000000000000000000000000000000000000000abcdabcd, z kolejki Q1 w menedżerze kolejek QM1:

```
curl -k "https://localhost:9443/ibmmq/rest/v3/messaging/qmgr/QM1/queue/Q1/message?
correlationId=ID:000000000000000000000000000000000000000000abcdabcd"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: text/plain"
```

USUŃ /messaging/qmgr/{qmgrName}/queue/{queueName}/message

Za pomocą metody DELETE HTTP z zasobem /messaging/qmgr/{qmgrName}/queue/{queueName}/message można destrukcyjnie pobrać komunikaty z powiązanego menedżera kolejek i kolejki.

Niszczy następny dostępny komunikat z określonego menedżera kolejek i kolejki, zwracając treść komunikatu w treści odpowiedzi HTTP . Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb. Komunikat musi mieć format MQSTR lub JMS TextMessagei jest odbierany przy użyciu bieżącego kontekstu użytkownika.

Niekompatybilne komunikaty pozostają w kolejce, a do programu wywołującego zwracany jest odpowiedni kod statusu. Na przykład komunikat, który nie ma formatu MQSTR lub JMS TextMessage .

V 9.3.0 Interfejs REST API V3 dodaje możliwość określania zdefiniowanych przez użytkownika właściwości komunikatu i dołączania priorytetu komunikatu do komunikatów. Nagłówki odpowiedzi ibm-mq-md-priority i ibm-mq-usr są dostępne tylko z interfejsem REST API V3. Nagłówek żądania ibm-mq-md-correlationId ma inny format w interfejsie REST API V3. Nagłówek może być identyfikatorem specyficznym dla aplikacji lub, jeśli jest to zakodowany łańcuch, zachowuje przedrostek ID: . Nagłówek odpowiedzi i parametr zapytania ibm-mq-md-messageId mają inny format w interfejsie REST API V3, zachowuje przedrostek ID: .

Adres URL materiałów

https://*host*:*port*/ibmmq/rest/v1/messaging/qmgr/*{qmgrName}*/queue/*{queueName}*/message

https://*host*:*port*/ibmmq/rest/v2/messaging/qmgr/*{qmgrName}*/queue/*{queueName}*/message

V 9.3.0 https://*host*:*port*/ibmmq/rest/v3/messaging/qmgr/*{qmgrName}*/queue/*{queueName}*/message

qmgrName

Określa nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie w celu przesyłania komunikatów. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb.

V 9.3.3 Z poziomu programu IBM MQ 9.3.3można nawiązać połączenie z lokalnym lub zdalnym menedżerem kolejek. Nazwa określona dla parametru **qmgrName** zależy od sposobu skonfigurowania serwera mqweb:

- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączenia tylko z lokalnymi menedżerami kolejek, należy użyć nazwy menedżera kolejek.

- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączeń ze zdalnymi menedżerami kolejek, należy użyć nazwy unikalnej dla menedżera kolejek zgodnie z informacjami o połączeniu ze zdalnym menedżerem kolejek.

Istnieje możliwość określenia, czy serwer mqweb jest skonfigurowany do nawiązywania połączeń z lokalnymi menedżerami kolejek, czy zdalnymi menedżerami kolejek, za pomocą komendy **dspmqweb properties** i wyświetlenia właściwości **mqRestMessagingConnectionMode** .

W nazwie menedżera kolejek jest rozróżniana wielkość liter.

Jeśli w nazwie menedżera kolejek znajdują się ukośniki, kropki lub znaki procentu, należy je zakodować w adresie URL:

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

queueName
Określa nazwę kolejki, z której zostanie otrzymany następny komunikat.

Kolejka musi być zdefiniowana jako lokalna lub jako alias wskazujący kolejkę lokalną.

W nazwie kolejki rozróżniana jest wielkość liter.

Jeśli nazwa kolejki zawiera ukośnik lub znak procentu, znaki te muszą być zakodowane jako URL :

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

Jeśli włączone są połączenia HTTP , można użyć protokołu HTTP zamiast protokołu HTTPS . Więcej informacji na temat włączania protokołu HTTPzawiera sekcja Konfigurowanie portów HTTP i HTTPS.

Opcjonalne parametry zapytania

REST API V2 REST API V1 **correlationId=*hexValue***
Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem korelacji.

hexValue
Parametr zapytania musi być określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
../message?correlationId=414d5120514d4144455620202020202067d8bf5923582e02
```

V 9.3.0 REST API V3 **correlationId= ID:*hexValue* lub correlationId=*wartość_specyfikacji_aplikacji***
Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem korelacji.

hexValue
Parametr zapytania musi być określony jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty i poprzedzony łańcuchem `"ID:"`.

Na przykład:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

wartość_specyfikacji_aplikacji
Parametr zapytania można określić jako łańcuch specyficzny dla aplikacji.

Na przykład:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=*hexValue***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem komunikatu.

hexValue

Parametr zapytania musi być określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
../message?messageId=414d5120514d414445562020202020067d8ce5923582f07
```

V 9.3.0 REST API V3 **messageId= ID:*hexValue***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem komunikatu.

hexValue

Parametr zapytania musi być określony jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty i poprzedzony łańcuchem `"ID:"`.

Na przykład:

```
../message?messageId=ID:414d5120514d414445562020202020067d8ce5923582f07
```

wait=*integerValue*

Określa, że metoda HTTP będzie oczekiwać *integerValue* ms na udostępnienie następnego komunikatu.

integerValue

Parametr zapytania musi być określony jako liczba całkowita reprezentująca milisekundowy przedział czasu. Wartość maksymalna to 2147483647.

Na przykład:

```
../message?wait=120000
```

Nagłówki żądań

Z żądaniem muszą być wysłane następujące nagłówki:

Autoryzacja

Ten nagłówek musi zostać wysłany, jeśli używane jest uwierzytelnianie podstawowe. Więcej informacji zawiera temat Korzystanie z podstawowego uwierzytelniania HTTP przy użyciu interfejsu REST API.

ibm-mq-rest-csrf-token

Ten nagłówek musi zostać wysłany. Może mieć dowolną wartość (nawet pustą).

Z żądaniem można opcjonalnie wysłać następujące nagłówki:

Akceptuj-zestaw znaków

Ten nagłówek może być używany do wskazywania, który zestaw znaków jest akceptowalny dla odpowiedzi. Jeśli ten nagłówek jest określony, musi być ustawiony jako UTF-8.

Accept-Language

Ten nagłówek określa wymagany język dla wszystkich wyjątków lub komunikatów o błędach zwracanych w treści komunikatu odpowiedzi.

Format treści żądania

Brak.

Wymogi dotyczące bezpieczeństwa

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb. Role `MQWebAdmin` i `MQWebAdminRO` nie mają zastosowania do messaging REST API. Więcej informacji na temat zabezpieczeń dla produktu REST API zawiera sekcja Zabezpieczenia IBM MQ Console i REST API.

Po uwierzytelnieniu na serwerze mqweb użytkownik może używać zarówno produktu messaging REST API , jak i produktu administrative REST API.

Użytkownik zabezpieczeń programu wywołującego musi mieć możliwość pobierania komunikatów z określonej kolejki:

- Kolejka określona przez część *{queueName}* adresu URL zasobu musi mieć włączoną opcję GET.
- MQ Appliance ALW W przypadku kolejki określonej przez część *{queueName}* adresu URL zasobu należy nadać uprawnienia +GET, +INQ i +BROWSE jednostce głównej zabezpieczeń programu wywołującego.
- z/OS W przypadku kolejki określonej przez część *{queueName}* adresu URL zasobu, UPDATE, należy zezwolić na dostęp jednostce głównej zabezpieczeń programu wywołującego.

Tą nazwą użytkownika zabezpieczeń może być użytkownik, który uruchomił serwer mqweb, lub użytkownik, który jest zalogowany na serwerze mqweb. Jeśli menedżer kolejek, z którym nawiązywane jest połączenie, jest zdalnym menedżerem kolejek, nazwą użytkownika zabezpieczeń może być użytkownik podany przez referencje w informacjach o połączeniu ze zdalnym menedżerem kolejek lub inny użytkownik określony przez reguły zabezpieczeń kanału. Więcej informacji na ten temat zawiera sekcja Określanie nazwy użytkownika używanej przez messaging REST API.

ALW W systemie AIX, Linux, and Windows można nadać uprawnienia użytkownikom zabezpieczeń, aby mogli korzystać z zasobów produktu IBM MQ, za pomocą komendy **setmqaut** . Więcej informacji zawiera temat **setmqaut** (nadawanie lub odbieranie uprawnienia).

z/OS W przypadku korzystania z systemu z/OS więcej informacji zawiera temat Konfigurowanie zabezpieczeń w systemie z/OS.

Kody statusu odpowiedzi

200
Komunikat został odebrany pomyślnie.

204
Brak komunikatu.

400
Podano niepoprawne dane.

Na przykład podano niepoprawną wartość parametru zapytania.

401
Nie uwierzytelniono.

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb i musi być elementem co najmniej jednej z ról `MQWebAdmin`, `MQWebAdminRO` lub `MQWebUser`. Należy również określić nagłówek `ibm-mq-rest-csrf-token` . Aby uzyskać więcej informacji, zapoznaj się z sekcją: “Wymogi dotyczące bezpieczeństwa” na stronie 2249.

403
Brak uprawnień.

Program wywołujący został uwierzytelniony na serwerze mqweb i jest powiązany z poprawną nazwą użytkownika. Jednak użytkownik nie ma dostępu do wszystkich lub podzbioru wymaganych zasobów IBM MQ albo nie ma dostępu do roli `MQWebUser` . Więcej informacji na temat wymaganego dostępu zawiera sekcja “Wymogi dotyczące bezpieczeństwa” na stronie 2249.

404
Kolejka nie istnieje.

405
Kolejka jest zablokowana metodą GET.

500
Problem z serwerem lub kod błędu z IBM MQ.

501
Nie można utworzyć odpowiedzi HTTP .

Na przykład odebrany komunikat ma niepoprawny typ lub ma poprawny typ, ale nie można przetworzyć treści.

502
Bieżąca nazwa użytkownika zabezpieczeń nie może odebrać komunikatu, ponieważ dostawca przesyłania komunikatów nie obsługuje wymaganej funkcji. Na przykład, jeśli ścieżka klasy serwera mqweb jest niepoprawna.

503
Menedżer kolejek nie jest uruchomiony.

Nagłówki odpowiedzi

Z odpowiedzią są zwracane następujące nagłówki:

Treść-język
Określa identyfikator języka komunikatu odpowiedzi w przypadku wystąpienia błędów lub wyjątków. Używany w połączeniu z nagłówkiem żądania `Accept-Language` w celu wskazania języka wymaganego dla warunków błędu lub wyjątku. Jeśli żądany język nie jest obsługiwany, używana jest wartość domyślna serwera mqweb.

Content-Length (długość treści)
Określa długość treści odpowiedzi HTTP , nawet jeśli nie ma treści. Wartość zawiera długość (w bajtach) danych komunikatu.

Content-Type
Określa typ treści zwracanej w treści odpowiedzi odebranego komunikatu. Po pomyślnym zakończeniu wartością jest `text/plain;charset=utf-8`. W przypadku wystąpienia błędów lub wyjątków wartością jest `application/json;charset=utf-8`.

REST API V2 REST API V1 **ibm-mq-md-correlationId**
Określa identyfikator korelacji odebranego komunikatu. Nagłówek jest zwracany, jeśli odebrany komunikat zawiera poprawny identyfikator korelacji. Jest on reprezentowany jako 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty.

Na przykład:

```
ibm-mq-md-correlationId: 414d5120514d4144455620202020202067d8bf5923582e02
```

V 9.3.0 REST API V3 **ibm-mq-md-correlationId**
Określa identyfikator korelacji odebranego komunikatu. Nagłówek jest zwracany, jeśli odebrany komunikat zawiera poprawny identyfikator korelacji. Identyfikator korelacji może mieć jedną z następujących postaci:

- 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty, poprzedzony łańcuchem `"ID:"`. Na przykład:

  ```
  ibm-mq-md-correlationId: ID:414d5120514d4144455620202020202067d8bf5923582e02
  ```

- Wartość specyficzna dla aplikacji. Wartość jest łańcuchem specyficznym dla aplikacji:

  ```
  ibm-mq-md-correlationId: My-Custom-CorrelId
  ```

ibm-mq-md-utrata ważności
Określa pozostały czas ważności odebranego komunikatu. Nagłówek może mieć jedną z następujących wartości:

Nieograniczony
Komunikat nie traci ważności.

Liczba całkowita
Pozostała liczba milisekund przed utratą ważności komunikatu.

REST API V2 REST API V1 **ibm-mq-md-messageId**
Określa identyfikator komunikatu, który jest przydzielany przez program IBM MQ do tego komunikatu. Podobnie jak nagłówek `ibm-mq-md-correlationId` , jest on reprezentowany jako 48-znakowy łańcuch zakodowany w formacie szesnastkowym, reprezentujący 24 bajty.

Na przykład:

```
ibm-mq-md-messageId: 414d5120514d4144556202020202067d8ce5923582f07
```

V 9.3.0 REST API V3 **ibm-mq-md-messageId**
Określa identyfikator komunikatu, który jest przydzielany przez program IBM MQ do tego komunikatu. Podobnie jak nagłówek `ibm-mq-md-correlationId` , jest on reprezentowany jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty, z przedrostkiem `"ID:"`

Na przykład:

```
ibm-mq-md-messageId: ID:414d5120514d4144556202020202067d8ce5923582f07
```

ibm-mq-md-persistence
Określa trwałość odebranego komunikatu. Nagłówek może mieć jedną z następujących wartości:

nonPersistent
Komunikat nie przetrwa awarii systemu lub restartów menedżera kolejek.

Trwały
Komunikat przetrwa awarie systemu lub restarty menedżera kolejek.

V 9.3.0 REST API V3 **ibm-mq-md-priorytet**
Zwraca ustawienie priorytetu komunikatu. Na przykład:

```
ibm-mq-md-priority: 3
```

ibm-mq-md-replyTo
Określa miejsce docelowe odpowiedzi dla odebranego komunikatu. W formacie nagłówka używana jest standardowa notacja kolejki odpowiedzi i menedżera kolejek `replyQueue@replyQmgr`.

Na przykład:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

V 9.3.3 **ibm-mq-rozwiązany-qmgr**
Określa nazwę menedżera kolejek, który został użyty dla żądania. Jeśli w zasobie URLużyto nazwy unikalnej, nagłówek ten identyfikuje nazwę menedżera kolejek, która jest powiązana z tą nazwą unikalną. Jeśli unikalna nazwa używana w zasobie URL odwołuje się do grupy menedżerów kolejek, ten nagłówek identyfikuje, który menedżer kolejek w grupie był używany.

V 9.3.0 REST API V3 **ibm-mq-usr**
Zwraca właściwości komunikatu zdefiniowane przez użytkownika. W komunikacie można ustawić wiele właściwości. W takim przypadku będą dwie lub więcej oddzielnych instancji nagłówka odpowiedzi ibm-mq-usr.

Na przykład:

```
ibm-mq-usr: myIProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp;true;boolean
```

Właściwości mają następującą składnię:

```
ibm-mq-usr: property_name; user_value; user_type
```

nazwa_właściwości
Nazwa określonej właściwości użytkownika. Musi to być poprawna nazwa właściwości JMS.

wartość_użytkownika
Wartość właściwości.

typ_użytkownika
Typ właściwości:

- `boolean` (prawda/fałsz, MQBOOL)
- `byte` (8-bitowa liczba całkowita, MQINT8)
- `short` (16-bitowa liczba całkowita, MQINT16)
- `integer` (32-bitowa liczba całkowita, MQINT32)
- `long` (64-bitowa liczba całkowita, MQINT64)
- `float` (32-bitowa liczba rzeczywista, MQFLOAT32)
- `double` (64-bitowa liczba rzeczywista, MQFLOAT64)
- `string` (łańcuch ujęty w cudzysłów)

Format treści odpowiedzi

Po pomyślnym zakończeniu treść odpowiedzi zawiera treść odebranego komunikatu. Jeśli wystąpi błąd, treść odpowiedzi zawiera sformatowany komunikat o błędzie JSON. Obie odpowiedzi są zakodowane jako UTF-8 . Więcej informacji na ten temat zawiera sekcja Obsługa błędów w systemieREST API.

Należy pamiętać, że podczas odbierania komunikatu obsługiwane są tylko komunikaty w formacie IBM MQ `MQSTR` i `JMS TextMessage` . Następnie wszystkie komunikaty są odbierane w punkcie synchronizacji, a wszystkie nieobsłużone komunikaty pozostają w kolejce. Kolejkę IBM MQ można skonfigurować w taki sposób, aby przenoś te komunikaty nieprzetwarzalne do alternatywnego miejsca docelowego. Więcej informacji na ten temat zawiera sekcja Obsługa komunikatów nieprzetwarzalnych w klasach IBM MQ classes for JMS.

Przykłady

W poniższych przykładach stosowany jest adres URL zasobu w wersji 2. Jeśli używana jest wersja produktu IBM MQ wcześniejsza niż IBM MQ 9.1.5, należy zamiast tego użyć adresu URL zasobu v1. W adresie URL zasobu z przykładu należy wpisać `v1` zamiast `v2`.

W poniższym przykładzie loguje się użytkownik o nazwie `mquser` z hasłem `mquser`. W przypadku komendy cURLżądanie logowania może wyglądać następująco: Windows . Znacznik LTPA jest zapisywany w pliku `cookiejar.txt` za pomocą opcji `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Po zalogowaniu się użytkownika znacznik LTPA i nagłówek `ibm-mq-rest-csrf-token` HTTP są używane do uwierzytelniania kolejnych żądań. `ibm-mq-rest-csrf-token` `token_value` może mieć dowolną wartość, w tym wartość pustą.

- Poniższy przykład komendy Windows cURL usuwa następny dostępny komunikat z kolejki Q1 w menedżerze kolejek QM1przy użyciu opcji domyślnych:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message"
-X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: text/plain"
```

- W poniższym przykładzie komendy Windows cURL usunięto komunikat o konkretnym identyfikatorze korelacji 000000000000000000000000000000000000000000abcdabcdz kolejki Q1 w menedżerze kolejek QM1:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
correlationId=000000000000000000000000000000000000000000abcdabcd"
-X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: text/plain"
```

- Poniższy przykład komendy Windows cURL usuwa komunikat o konkretnym identyfikatorze korelacji 000000000000000000000000000000000000000000abcdabcdz kolejki Q1 w menedżerze kolejek QM1, czekając maksymalnie 30 sekund na udostępnienie komunikatu. Jeśli minie 30 sekund bez umieszczenia określonego komunikatu w kolejce, wywołanie DELETE zwróci komunikat:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/message?
correlationId=000000000000000000000000000000000000000000abcdabcd&wait=30000"
-X DELETE -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: text/plain"
```

/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist

Aby uzyskać listę dostępnych komunikatów z określonej kolejki w określonym menedżerze kolejek, można użyć metody GET HTTP z zasobem /messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist .

GET /messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist

Aby uzyskać listę dostępnych komunikatów z określonej kolejki w określonym menedżerze kolejek, można użyć metody GET HTTP z zasobem /messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist .

Przegląda listę podsumowania komunikatów z określonego menedżera kolejek i kolejki. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb. Dane podsumowania są zwracane w treści odpowiedzi HTTP w postaci tablicy w formacie JSON. Dane nie zawierają ładunku komunikatów i są odbierane przy użyciu bieżącego kontekstu użytkownika. Żadne komunikaty nie są usuwane z powiązanej kolejki.

Jeśli zostanie wysłane żądanie pobrania listy dostępnych komunikatów z kolejki, która jest zablokowana za pomocą komendy GET, zwracana jest pusta tablica JSON.

- "Adres URL materiałów" na stronie 2254
- "Opcjonalne parametry zapytania" na stronie 2254
- "Nagłówki żądań" na stronie 2255
- "Format treści żądania" na stronie 2256
- "Wymogi dotyczące bezpieczeństwa" na stronie 2256
- "Kody statusu odpowiedzi" na stronie 2256
- "Nagłówki odpowiedzi" na stronie 2257
- "Format treści odpowiedzi" na stronie 2257
- "Przykłady" na stronie 2258

Adres URL materiałów

`https://host:port/ibmmq/rest/v1/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

`https://host:port/ibmmq/rest/v2/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

V 9.3.0 `https://host:port/ibmmq/rest/v3/messaging/qmgr/{qmgrName}/queue/{queueName}/messagelist`

qmgrName
Określa nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie w celu przesyłania komunikatów. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb.

V 9.3.3 Z poziomu programu IBM MQ 9.3.3można nawiązać połączenie z lokalnym lub zdalnym menedżerem kolejek. Nazwa określona dla parametru **qmgrName** zależy od sposobu skonfigurowania serwera mqweb:

- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączenia tylko z lokalnymi menedżerami kolejek, należy użyć nazwy menedżera kolejek.
- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączeń ze zdalnymi menedżerami kolejek, należy użyć nazwy unikalnej dla menedżera kolejek zgodnie z informacjami o połączeniu ze zdalnym menedżerem kolejek.

Istnieje możliwość określenia, czy serwer mqweb jest skonfigurowany do nawiązywania połączeń z lokalnymi menedżerami kolejek, czy zdalnymi menedżerami kolejek, za pomocą komendy **dspmqweb properties** i wyświetlenia właściwości **mqRestMessagingConnectionMode** .

W nazwie menedżera kolejek jest rozróżniana wielkość liter.

Jeśli w nazwie menedżera kolejek znajdują się ukośniki, kropki lub znaki procentu, należy je zakodować w adresie URL:

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

queueName
Określa nazwę kolejki, z której mają być przeglądane komunikaty.

Kolejka musi być zdefiniowana jako lokalna lub jako alias wskazujący kolejkę lokalną.

W nazwie kolejki rozróżniana jest wielkość liter.

Jeśli nazwa kolejki zawiera ukośnik lub znak procentu, znaki te muszą być zakodowane jako URL :

- Ukośnik (/) musi być zakodowany jako %2F.
- Znak procentu (%) musi być zakodowany jako %25.

Jeśli włączone są połączenia HTTP , można użyć protokołu HTTP zamiast protokołu HTTPS . Więcej informacji na temat włączania protokołu HTTPzawiera sekcja Konfigurowanie portów HTTP i HTTPS.

Opcjonalne parametry zapytania

REST API V2 **correlationId=*hexValue***
Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem korelacji.

hexValue
Parametr zapytania musi być określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
../messagelist?correlationId=414d5120514d414445562020202020202067d8bf5923582e02
```

V 9.3.0 REST API V3 **correlationId= ID:*hexValue* lub correlationId=*wartość_specyfikacji_aplikacji***

Określa, że metoda HTTP zwraca listę komunikatów z odpowiednim identyfikatorem korelacji.

hexValue

Parametr zapytania musi być określony jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty i poprzedzony łańcuchem `"ID:"`.

Na przykład:

```
../message?correlationId=ID:414d5120514d4144455620202020202067d8bf5923582e02
```

wartość_specyfikacji_aplikacji

Parametr zapytania można określić jako łańcuch specyficzny dla aplikacji.

Na przykład:

```
../message?correlationId=My-Custom-CorrelId
```

REST API V2 REST API V1 **messageId=*hexValue***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem komunikatu.

hexValue

Parametr zapytania musi być określony jako 48-znakowy łańcuch zakodowany szesnastkowo, reprezentujący 24 bajty.

Na przykład:

```
../message?messageId=414d5120514d4144455620202020202067d8ce5923582f07
```

V 9.3.0 REST API V3 **messageId= ID:*hexValue***

Określa, że metoda HTTP zwraca następny komunikat z odpowiednim identyfikatorem komunikatu.

hexValue

Parametr zapytania musi być określony jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty i poprzedzony łańcuchem `"ID:"`.

Na przykład:

```
../message?messageId=ID:414d5120514d4144455620202020202067d8ce5923582f07
```

limit=*integerValue*

Określa, że treść odpowiedzi metody HTTP jest ograniczona do elementów JSON *integerValue* .

integerValue

Parametr zapytania musi być określony jako liczba całkowita reprezentująca maksymalną liczbę elementów zawartych w treści odpowiedzi JSON.

Wartością domyślną jest 10, a wartością maksymalną 2147483647.

Na przykład:

```
../messagelist?limit=250
```

Nagłówki żądań

Z żądaniem muszą być wysłane następujące nagłówki:

Autoryzacja

Ten nagłówek musi zostać wysłany, jeśli używane jest uwierzytelnianie podstawowe. Więcej informacji zawiera temat Korzystanie z podstawowego uwierzytelniania HTTP przy użyciu interfejsu REST API.

ibm-mq-rest-csrf-token
Ten nagłówek musi zostać wysłany. Może mieć dowolną wartość (nawet pustą).

Z żądaniem można opcjonalnie wysłać następujące nagłówki:

Akceptuj-zestaw znaków
Ten nagłówek może być używany do wskazywania, który zestaw znaków jest akceptowalny dla odpowiedzi. Jeśli ten nagłówek jest określony, musi być ustawiony jako UTF-8.

Accept-Language
Ten nagłówek określa wymagany język dla wszystkich wyjątków lub komunikatów o błędach zwracanych w treści komunikatu odpowiedzi.

Format treści żądania

Brak.

Wymogi dotyczące bezpieczeństwa

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb. Role `MQWebAdmin` i `MQWebAdminRO` nie mają zastosowania do messaging REST API. Więcej informacji na temat zabezpieczeń dla produktu REST API zawiera sekcja Zabezpieczenia IBM MQ Console i REST API.

Po uwierzytelnieniu na serwerze mqweb użytkownik może używać zarówno produktu messaging REST API , jak i produktu administrative REST API.

Nazwa użytkownika zabezpieczeń osoby nawiązującej połączenie musi mieć przypisane uprawnienie umożliwiające przeglądanie komunikatów z określonej kolejki:

- Kolejka określona przez część *{queueName}* adresu URL zasobu musi mieć włączoną opcję `BROWSE`.
- ALW MQ Appliance W przypadku kolejki określonej przez część *{queueName}* adresu URL zasobu należy nadać uprawnienia `+GET`, `+INQ` i `+BROWSE` jednostce głównej zabezpieczeń programu wywołującego.
- z/OS W przypadku kolejki określonej przez część *{queueName}* adresu URL zasobu, `UPDATE`, należy zezwolić na dostęp jednostce głównej zabezpieczeń programu wywołującego.

Tą nazwą użytkownika zabezpieczeń może być użytkownik, który uruchomił serwer mqweb, lub użytkownik, który jest zalogowany na serwerze mqweb. Jeśli menedżer kolejek, z którym nawiązywane jest połączenie, jest zdalnym menedżerem kolejek, nazwą użytkownika zabezpieczeń może być użytkownik podany przez referencje w informacjach o połączeniu ze zdalnym menedżerem kolejek lub inny użytkownik określony przez reguły zabezpieczeń kanału. Więcej informacji na ten temat zawiera sekcja Określanie nazwy użytkownika używanej przez messaging REST API.

ALW W systemie AIX, Linux, and Windows można nadać uprawnienia użytkownikom zabezpieczeń, aby mogli korzystać z zasobów produktu IBM MQ, za pomocą komendy **setmqaut** . Więcej informacji zawiera temat **setmqaut** (nadawanie lub odbieranie uprawnienia).

z/OS W przypadku korzystania z systemu z/OS więcej informacji zawiera temat Konfigurowanie zabezpieczeń w systemie z/OS.

Kody statusu odpowiedzi

200
Lista komunikatów została pomyślnie odebrana.

400
Podano niepoprawne dane.

Na przykład podano niepoprawną wartość parametru zapytania.

401
Nie uwierzytelniono.

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb i musi być elementem co najmniej jednej z ról `MQWebAdmin`, `MQWebAdminRO` lub `MQWebUser`. Należy również określić nagłówek `ibm-mq-rest-csrf-token` . Aby uzyskać więcej informacji, zapoznaj się z sekcją: "Wymogi dotyczące bezpieczeństwa" na stronie 2256.

403
Brak uprawnień.

Program wywołujący został uwierzytelniony na serwerze mqweb i jest powiązany z poprawną nazwą użytkownika. Jednak użytkownik nie ma dostępu do wszystkich lub podzbioru wymaganych zasobów IBM MQ albo nie ma dostępu do roli `MQWebUser` . Więcej informacji na temat wymaganego dostępu zawiera sekcja "Wymogi dotyczące bezpieczeństwa" na stronie 2256.

404
Kolejka nie istnieje.

500
Problem z serwerem lub kod błędu z IBM MQ.

501
Nie można utworzyć odpowiedzi HTTP .

Na przykład odebrany komunikat ma niepoprawny typ lub ma poprawny typ, ale nie można przetworzyć treści.

502
Bieżąca nazwa użytkownika zabezpieczeń nie może odebrać komunikatu, ponieważ dostawca przesyłania komunikatów nie obsługuje wymaganej funkcji. Na przykład, jeśli ścieżka klasy serwera mqweb jest niepoprawna.

503
Menedżer kolejek nie jest uruchomiony.

Nagłówki odpowiedzi

Treść-język
Określa identyfikator języka komunikatu odpowiedzi w przypadku wystąpienia błędów lub wyjątków. Używany w połączeniu z nagłówkiem żądania `Accept-Language` w celu wskazania języka wymaganego dla warunków błędu lub wyjątku. Jeśli żądany język nie jest obsługiwany, używana jest wartość domyślna serwera mqweb.

Content-Length (długość treści)
Określa długość treści odpowiedzi HTTP , nawet jeśli nie ma treści. Wartość zawiera długość danych komunikatu w bajtach.

Content-Type
Określa typ treści odpowiedzi. Wartością jest `application/json;charset=utf-8`.

V 9.3.3 **ibm-mq-rozwiązany-qmgr**
Określa nazwę menedżera kolejek, który został użyty dla żądania. Jeśli w zasobie URLużyto nazwy unikalnej, nagłówek ten identyfikuje nazwę menedżera kolejek, która jest powiązana z tą nazwą unikalną. Jeśli unikalna nazwa używana w zasobie URL odwołuje się do grupy menedżerów kolejek, ten nagłówek identyfikuje, który menedżer kolejek w grupie był używany.

Format treści odpowiedzi

Po pomyślnym zakończeniu działania treść odpowiedzi jest zakodowaną odpowiedzią UTF-8 . Odpowiedź zawiera zewnętrzny obiekt JSON, który zawiera pojedynczą tablicę JSON o nazwie `messages`. Każdy element w tablicy jest obiektem JSON, który zawiera informacje o komunikacie w kolejce. Każdy element zawiera następujące atrybuty:

REST API V2 REST API V1 **correlationId**
Określa identyfikator korelacji komunikatu. Wartość jest zwracana, jeśli komunikat zawiera poprawny identyfikator korelacji.

V 9.3.0 RESTAPI V3**correlationId**
Określa identyfikator korelacji komunikatu. Wartość jest zwracana, jeśli komunikat zawiera poprawny identyfikator korelacji. Identyfikator korelacji jest poprzedzony łańcuchem "ID:" lub może być wartością specyficzną dla aplikacji.

RESTAPI V2 RESTAPI V1**messageId**
Określa identyfikator komunikatu, który jest przydzielany przez produkt IBM MQ do tego komunikatu. Jest on reprezentowany jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty.

V 9.3.0 RESTAPI V3**messageId**
Określa identyfikator komunikatu, który jest przydzielany przez produkt IBM MQ do tego komunikatu. Jest on reprezentowany jako 48-znakowy zakodowany łańcuch szesnastkowy, reprezentujący 24 bajty. Identyfikator komunikatu jest poprzedzony łańcuchem "ID:".

format
Określa pole formatu MQMD. W normalnych okolicznościach komunikaty tekstowe będą zawierać wartość IBM MQ MQSTR .

Jeśli zostanie wysłane żądanie pobrania listy komunikatów w kolejce, która jest zablokowana za pomocą komendy GET, zwracana jest pusta tablica JSON.

Jeśli wystąpi błąd, treść odpowiedzi zawiera sformatowany komunikat o błędzie JSON. Więcej informacji na ten temat zawiera sekcja Obsługa błędów w systemieREST API.

Przykłady

W poniższych przykładach stosowany jest adres URL zasobu w wersji 2. Jeśli używana jest wersja produktu IBM MQ wcześniejsza niż IBM MQ 9.1.5, należy zamiast tego użyć adresu URL zasobu v1. W adresie URL zasobu z przykładu należy wpisać v1 zamiast v2.

W poniższym przykładzie loguje się użytkownik o nazwie `mquser` z hasłem `mquser`. W przypadku komendy cURLżądanie logowania może wyglądać następująco: Windows . Znacznik LTPA jest zapisywany w pliku `cookiejar.txt` za pomocą opcji `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v2/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Po zalogowaniu się użytkownika znacznik LTPA i nagłówek `ibm-mq-rest-csrf-token` HTTP są używane do uwierzytelniania kolejnych żądań. `ibm-mq-rest-csrf-token` `token_value` może mieć dowolną wartość, w tym wartość pustą.

- W poniższym przykładzie komendy Windows cURL przedstawiono dziesięć następnych dostępnych komunikatów z kolejki Q1 w menedżerze kolejek QM1z użyciem opcji domyślnych:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: application/json"
  ```

- W poniższym przykładzie komendy Windows cURL przedstawiono listę następnych dwieście dostępnych komunikatów z kolejki Q1 w menedżerze kolejek QM1z użyciem opcji domyślnych:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?
  limit=200"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: application/json"
  ```

- RESTAPI V2 RESTAPI V1W poniższym przykładzie komendy Windows cURL wyświetlane są tylko te komunikaty, które mają odpowiedni identyfikator korelacji

000000000000000000000000000000000000000000abcdabcd, z kolejki Q1 w menedżerze kolejek QM1:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/queue/Q1/messagelist?
correlationId=000000000000000000000000000000000000000000abcdabcd"
-X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
-H "Accept: application/json"
```

- V 9.3.0 REST API V3

  Poniższy przykład komendy Windows cURL jest taki sam jak w poprzednim przykładzie, ale używa interfejsu REST API V3. W przykładzie wyświetlane są tylko te komunikaty, które mają odpowiedni identyfikator korelacji, 000000000000000000000000000000000000000000abcdabcd, z kolejki Q1 w menedżerze kolejek QM1:

  ```
  curl -k "https://localhost:9443/ibmmq/rest/v3/messaging/qmgr/QM1/queue/Q1/messagelist?
  correlationId=ID:000000000000000000000000000000000000000000abcdabcd"
  -X GET -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token-value"
  -H "Accept: application/json"
  ```

/messaging/qmgr/{qmgrName}/topic/{topicString}/message

Do publikowania komunikatów w określonym temacie w określonym menedżerze kolejek można użyć metody HTTP POST z zasobem /messaging/qmgr/{qmgrName}/topic/{topicString}/message .

POST /messaging/qmgr/{qmgrName}/topic/{topicString}/message

Do publikowania komunikatów w określonym temacie w określonym menedżerze kolejek można użyć metody HTTP POST z zasobem /messaging/qmgr/{qmgrName}/topic/{topicString}/message .

Publikuje komunikat tekstowy w treści żądania HTTP w określonym menedżerze kolejek i temacie. Menedżer kolejek musi znajdować się na tym samym komputerze co serwer mqweb i obsługiwane są tylko komunikaty tekstowe. Komunikaty są publikowane w formacie MQSTR lub JMS TextMessage przy użyciu bieżącego kontekstu użytkownika i mają domyślny priorytet 4 dla interfejsu REST API V2oraz asDestination dla interfejsu REST API V3.

V 9.3.0 Interfejs REST API V3 dodaje możliwość określenia zdefiniowanych przez użytkownika właściwości komunikatu i uwzględnienia priorytetu komunikatu. Nagłówki żądań ibm-mq-md-priority i ibm-mq-usr są dostępne tylko z interfejsem REST API V3. Nagłówek żądania ibm-mq-md-correlationId ma inny format w interfejsie REST API V3. Nagłówek może być identyfikatorem specyficznym dla aplikacji lub, jeśli zakodowany łańcuch wymaga przedrostka ID: . Jeśli żądanie POST zawiera komunikaty zdefiniowane przez użytkownika lub identyfikator korelacji specyficzny dla aplikacji, komunikat jest sformatowany jako JMS TextMessage.

Adres URL materiałów

https://*host*:*port*/ibmmq/rest/v2/messaging/qmgr/*{qmgrName}*/topic/*{topicString}*/message

V 9.3.0 https://*host*:*port*/ibmmq/rest/v3/messaging/qmgr/*{qmgrName}*/topic/*{topicString}*/message

qmgrName
Określa nazwę menedżera kolejek, z którym ma zostać nawiązane połączenie w celu przesyłania komunikatów.

V 9.3.3 Z poziomu programu IBM MQ 9.3.3można nawiązać połączenie z lokalnym lub zdalnym menedżerem kolejek. Nazwa określona dla parametru **qmgrName** zależy od sposobu skonfigurowania serwera mqweb:

- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączenia tylko z lokalnymi menedżerami kolejek, należy użyć nazwy menedżera kolejek.
- Jeśli serwer mqweb jest skonfigurowany do nawiązywania połączeń ze zdalnymi menedżerami kolejek, należy użyć nazwy unikalnej dla menedżera kolejek zgodnie z informacjami o połączeniu ze zdalnym menedżerem kolejek.

Istnieje możliwość określenia, czy serwer mqweb jest skonfigurowany do nawiązywania połączeń z lokalnymi menedżerami kolejek, czy zdalnymi menedżerami kolejek, za pomocą komendy **dspmqweb properties** i wyświetlenia właściwości **mqRestMessagingConnectionMode** .

W nazwie uwzględniana jest wielkość liter.

Jeśli nazwa zawiera ukośnik, kropkę lub znak procentu, następujące znaki muszą być zakodowane przy użyciu kodowania URL :

- Ukośnik musi być zakodowany jako %2F.
- Kropka musi być zakodowana jako %2E.
- Znak procentu musi być zakodowany jako %25.

topicString
Określa łańcuch tematu, dla którego ma zostać opublikowany komunikat.

W łańcuchu tematu rozróżniana jest wielkość liter. Łańcuch tematu może zawierać wiele poziomów tematu oddzielonych separatorem ukośnika.

Jeśli łańcuch tematu zawiera znak procentu, kropkę lub znak zapytania, następujące znaki muszą być zakodowane przy użyciu kodowania URL :

- Znak procentu musi być zakodowany jako %25.
- Kropka musi być zakodowana jako %2E.
- Znak zapytania musi być zakodowany jako %3F.

Jeśli łańcuch tematu rozpoczyna się lub kończy ukośnikiem, musi być zakodowany przy użyciu znaku %2F.

Na przykład, aby opublikować w łańcuchu tematu:

- `sport/football` w menedżerze kolejek `MY.QMGR`należy użyć następującego URL:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/sport/football/message
```

- `/sport/football` w menedżerze kolejek `MY.QMGR`należy użyć następującego URL:

```
https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/MY%2EQMGR/topic/%2Fsport/football/
message
```

Jeśli włączone są połączenia HTTP , można użyć protokołu HTTP zamiast protokołu HTTPS . Więcej informacji na temat włączania protokołu HTTPzawiera sekcja Konfigurowanie portów HTTP i HTTPS.

Nagłówki żądań

Z żądaniem muszą być wysłane następujące nagłówki:

Autoryzacja
Ten nagłówek musi zostać wysłany, jeśli używane jest uwierzytelnianie podstawowe. Więcej informacji zawiera temat Korzystanie z podstawowego uwierzytelniania HTTP przy użyciu interfejsu REST API.

Content-Type
Ten nagłówek musi być wysłany z jedną z następujących wartości:

- `text/plain;charset=utf-8`
- `text/html;charset=utf-8`
- `text/xml;charset=utf-8`
- `application/json;charset=utf-8`
- `application/xml;charset=utf-8`

ibm-mq-rest-csrf-token
Ten nagłówek musi zostać wysłany. Może mieć dowolną wartość (nawet pustą).

Z żądaniem można opcjonalnie wysłać następujące nagłówki:

Accept-Language
Ten nagłówek określa wymagany język dla wszystkich wyjątków lub komunikatów o błędach zwracanych w treści komunikatu odpowiedzi.

ibm-mq-md-utrata ważności
Ten nagłówek ustawia czas ważności dla utworzonego komunikatu. Utrata ważności komunikatu rozpoczyna się od momentu nadejścia komunikatu do menedżera kolejek. W wyniku tego opóźnienie sieci jest ignorowane. Nagłówek musi być określony jako jedna z następujących wartości:

Nieograniczony
Komunikat nie traci ważności.

Jest to wartość domyślna.

Liczba całkowita
Milisekundy przed utratą ważności komunikatu.

Ograniczone do zakresu od 0 do 99999999900.

ibm-mq-md-persistence
Ten nagłówek ustawia trwałość dla utworzonego komunikatu. Nagłówek musi być określony jako jedna z następujących wartości:

nonPersistent
Komunikat nie przetrwa awarii systemu lub restartów menedżera kolejek.

Jest to wartość domyślna.

Trwały
Komunikat przetrwa awarie systemu lub restarty menedżera kolejek.

ibm-mq-md-priorytet
Ten nagłówek ustawia priorytet utworzonego komunikatu. Nagłówek musi być określony jako jedna z następujących wartości:

asDestination
Komunikat używa priorytetu określonego w atrybucie DEFPRTY bazowego obiektu kolejki IBM MQ .

Liczba całkowita
Określ rzeczywisty priorytet jako liczbę całkowitą z zakresu od 0 do 9.

Na przykład:

```
ibm-mq-md-priority: asDestination
```

ibm-mq-md-replyTo

Ten nagłówek ustawia miejsce docelowe odpowiedzi dla utworzonego komunikatu. Format nagłówka używa standardowej notacji dostarczania kolejki odpowiedzi i opcjonalnego menedżera kolejek: `replyQueue[@replyQmgr]`

Na przykład:

```
ibm-mq-md-replyTo: myReplyQueue@myReplyQMgr
```

V 9.3.0 REST API V3 **ibm-mq-usr**

Ustaw zdefiniowane przez użytkownika właściwości komunikatu żądania. W komunikacie można ustawić wiele właściwości. W pojedynczym nagłówku żądania ibm-mq-usr można podać wiele właściwości rozdzielonych przecinkami lub można użyć co najmniej dwóch oddzielnych instancji nagłówka żądania ibm-mq-usr.

Na przykład:

```
ibm-mq-usr: myIProp;5;short
ibm-mq-usr: mySProp;"hi";string
ibm-mq-usr: myBProp;true;boolean
ibm-mq-usr: myA;5;byte,myB;-10;integer
```

Właściwości mają następującą składnię:

```
ibm-mq-usr: property_name; user_value; user_type
```

nazwa_właściwości

Nazwa określonej właściwości użytkownika. Musi to być poprawna nazwa właściwości JMS.

wartość_użytkownika

Wartość właściwości.

typ_użytkownika

Typ właściwości:

- `boolean` (prawda/fałsz, MQBOOL)
- `byte` (8-bitowa liczba całkowita, MQINT8)
- `short` (16-bitowa liczba całkowita, MQINT16)
- `integer` (32-bitowa liczba całkowita, MQINT32)
- `long` (64-bitowa liczba całkowita, MQINT64)
- `float` (32-bitowa liczba rzeczywista, MQFLOAT32)
- `double` (64-bitowa liczba rzeczywista, MQFLOAT64)
- `string` (łańcuch ujęty w cudzysłów)

Format treści żądania

Treść żądania musi być tekstem i musi być zakodowana w formacie UTF-8 . Nie jest wymagana żadna konkretna struktura tekstowa. Sformatowany komunikat MQSTR zawierający treść żądania jest tworzony i publikowany w określonym temacie.

V 9.3.0 REST API V3 Jeśli używane są zdefiniowane przez użytkownika właściwości interfejsu REST API w wersji V3 lub specyficzne dla aplikacji funkcje identyfikatora korelacji, tworzony jest sformatowany komunikat JMS `TextMessage` zawierający treść żądania i umieszczany w określonej kolejce.

Więcej informacji na ten temat zawierają przykłady.

Wymogi dotyczące bezpieczeństwa

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb. Role `MQWebAdmin` i `MQWebAdminRO` nie mają zastosowania do messaging REST API. Więcej informacji na temat zabezpieczeń dla produktu REST API zawiera sekcja Zabezpieczenia IBM MQ Console i REST API.

Po uwierzytelnieniu na serwerze mqweb użytkownik może używać zarówno produktu messaging REST API , jak i produktu administrative REST API.

Użytkownik zabezpieczeń programu wywołującego musi mieć możliwość publikowania komunikatów w określonym temacie:

- Temat określony w części *{topicString}* zasobu URL musi mieć włączoną opcję `PUBLISH` .
- MQ Appliance ALW W przypadku tematu określonego przez część *{topicString}* zasobu URLnależy nadać uprawnienie +PUB użytkownikowi zabezpieczeń programu wywołującego.
- z/OS W przypadku tematu określonego przez część *{topicString}* zasobu URLdostęp `UPDATE` musi zostać nadany użytkownikowi zabezpieczeń programu wywołującego.

Tą nazwą użytkownika zabezpieczeń może być użytkownik, który uruchomił serwer mqweb, lub użytkownik, który jest zalogowany na serwerze mqweb. Jeśli menedżer kolejek, z którym nawiązywane jest połączenie, jest zdalnym menedżerem kolejek, nazwą użytkownika zabezpieczeń może być użytkownik podany przez referencje w informacjach o połączeniu ze zdalnym menedżerem kolejek lub inny użytkownik określony przez reguły zabezpieczeń kanału. Więcej informacji na ten temat zawiera sekcja Określanie nazwy użytkownika używanej przez messaging REST API.

ALW W systemie AIX, Linux, and Windows można nadać uprawnienia użytkownikom zabezpieczeń, aby mogli korzystać z zasobów produktu IBM MQ, za pomocą komendy **`setmqaut`** . Więcej informacji zawiera temat **`setmqaut`** (nadawanie lub odbieranie uprawnienia).

z/OS W przypadku korzystania z systemu z/OS więcej informacji zawiera temat Konfigurowanie zabezpieczeń w systemie z/OS.

Jeśli z serwerem messaging REST APIużywane są zaawansowane zabezpieczenia komunikatów (Advanced Message Security-AMS), należy pamiętać, że wszystkie komunikaty są szyfrowane przy użyciu kontekstu serwera mqweb, a nie kontekstu użytkownika, który opublikuje komunikat.

Kody statusu odpowiedzi

201
Komunikat został pomyślnie utworzony i opublikowany.

400
Podano niepoprawne dane.

Na przykład podano niepoprawną wartość nagłówka żądania.

401
Nie uwierzytelniono.

Program wywołujący musi zostać uwierzytelniony na serwerze mqweb i musi być elementem co najmniej jednej z ról `MQWebAdmin`, `MQWebAdminRO` lub `MQWebUser`. Należy również określić nagłówek `ibm-mq-rest-csrf-token` . Aby uzyskać więcej informacji, zapoznaj się z sekcją: “Wymogi dotyczące bezpieczeństwa” na stronie 2263.

403
Brak uprawnień.

Program wywołujący został uwierzytelniony na serwerze mqweb i jest powiązany z poprawną nazwą użytkownika. Jednak użytkownik nie ma dostępu do wszystkich lub podzbioru wymaganych zasobów IBM MQ albo nie ma dostępu do roli `MQWebUser` . Więcej informacji na temat wymaganego dostępu zawiera sekcja “Wymogi dotyczące bezpieczeństwa” na stronie 2263.

404
Menedżer kolejek nie istnieje.

405
Publikowanie tematu jest zablokowane.

415
Nagłówek lub treść komunikatu jest nieobsługiwanym typem nośnika.

Na przykład nagłówek `Content-Type` jest ustawiony na nieobsługiwany typ nośnika.

500
Problem z serwerem lub kod błędu z IBM MQ.

502
Bieżąca nazwa użytkownika zabezpieczeń nie może opublikować komunikatu, ponieważ dostawca przesyłania komunikatów nie obsługuje wymaganej funkcji. Na przykład, jeśli ścieżka klasy serwera mqweb jest niepoprawna.

503
Menedżer kolejek nie jest uruchomiony.

Nagłówki odpowiedzi

Z odpowiedzią są zwracane następujące nagłówki:

Treść-język
Określa identyfikator języka komunikatu odpowiedzi w przypadku wystąpienia błędów lub wyjątków. Używany w połączeniu z nagłówkiem żądania `Accept-Language` w celu wskazania języka wymaganego dla warunków błędu lub wyjątku. Jeśli żądany język nie jest obsługiwany, używana jest wartość domyślna serwera mqweb.

Content-Length (długość treści)
Określa długość treści odpowiedzi HTTP , nawet jeśli nie ma treści. Po pomyślnym zakończeniu wartość wynosi zero.

Content-Type
Określa typ treści odpowiedzi. Po pomyślnym zakończeniu wartością jest `text/plain;charset=utf-8`. W przypadku wystąpienia błędów lub wyjątków wartością jest `application/json;charset=utf-8`.

V 9.3.3 **ibm-mq-rozwiązany-qmgr**
Określa nazwę menedżera kolejek, który został użyty dla żądania. Jeśli w zasobie URLużyto nazwy unikalnej, nagłówek ten identyfikuje nazwę menedżera kolejek, która jest powiązana z tą nazwą unikalną. Jeśli unikalna nazwa używana w zasobie URL odwołuje się do grupy menedżerów kolejek, ten nagłówek identyfikuje, który menedżer kolejek w grupie był używany.

Format treści odpowiedzi

Treść odpowiedzi jest pusta, jeśli komunikat został pomyślnie opublikowany. Jeśli wystąpi błąd, treść odpowiedzi zawiera komunikat o błędzie. Więcej informacji na ten temat zawiera sekcja Obsługa błędów w systemieREST API.

Przykłady

W poniższym przykładzie loguje się użytkownik o nazwie `mquser` z hasłem `mquser`. W przypadku komendy cURLżądanie logowania może wyglądać następująco: Windows . Znacznik LTPA jest zapisywany w pliku `cookiejar.txt` za pomocą opcji `-c` :

```
curl -k "https://localhost:9443/ibmmq/rest/v1/login" -X POST
-H "Content-Type: application/json" --data "{\"username\":\"mquser\",\"password\":\"mquser\"}"
-c c:\cookiejar.txt
```

Po zalogowaniu się użytkownika znacznik LTPA i nagłówek `ibm-mq-rest-csrf-token` HTTP są używane do uwierzytelniania kolejnych żądań. `ibm-mq-rest-csrf-token token_value` może mieć dowolną wartość, w tym wartość pustą.

- Poniższy przykład komendy Windows cURL powoduje opublikowanie komunikatu w łańcuchu tematu myTopic w menedżerze kolejek QM1przy użyciu opcji domyślnych. Komunikat zawiera tekst *"Hello World!"*:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic/message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain;charset=utf-8" --data "Hello World!"
```

- Poniższy przykład komendy Windows cURL powoduje opublikowanie trwałego komunikatu w łańcuchu tematu myTopic/thisTopic w menedżerze kolejek QM1z utratą ważności 2 minut. Komunikat zawiera tekst *"Hello World!"*:

```
curl -k "https://localhost:9443/ibmmq/rest/v2/messaging/qmgr/QM1/topic/myTopic%2FthisTopic/
message"
-X POST -b c:\cookiejar.txt -H "ibm-mq-rest-csrf-token: token_value"
-H "Content-Type: text/plain;charset=utf-8" -H "ibm-mq-md-persistence: persistent"
-H "ibm-mq-md-expiry: 120000" --data "Hello World!"
```

Uwagi

Niniejsza publikacja została opracowana z myślą o produktach i usługach oferowanych w Stanach Zjednoczonych.

IBM może nie oferować w innych krajach produktów, usług lub opcji omawianych w tej publikacji. Informacje o produktach i usługach dostępnych w danym kraju można uzyskać od lokalnego przedstawiciela IBM. Odwołanie do produktu, programu lub usługi IBM nie oznacza, że można użyć wyłącznie tego produktu, programu lub usługi IBM. Zamiast nich można zastosować ich odpowiednik funkcjonalny pod warunkiem, że nie narusza to praw własności intelektualnej firmy IBM. Jednakże cała odpowiedzialność za ocenę przydatności i sprawdzenie działania produktu, programu lub usługi pochodzących od producenta innego niż IBM spoczywa na użytkowniku.

IBM może posiadać patenty lub złożone wnioski patentowe na towary i usługi, o których mowa w niniejszej publikacji. Przedstawienie niniejszej publikacji nie daje żadnych uprawnień licencyjnych do tychże patentów. Pisemne zapytania w sprawie licencji można przesyłać na adres:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Zapytania w sprawie licencji dotyczących informacji kodowanych przy użyciu dwubajtowych zestawów znaków (DBCS) należy kierować do lokalnych działów IBM Intellectual Property Department lub zgłaszać na piśmie pod adresem:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

Poniższy akapit nie obowiązuje w Wielkiej Brytanii, a także w innych krajach, w których jego treść pozostaje w sprzeczności z przepisami prawa miejscowego: INTERNATIONAL BUSINESS MACHINES CORPORATION DOSTARCZA TĘ PUBLIKACJĘ W STANIE, W JAKIM SIĘ ZNAJDUJE ("AS IS"), BEZ JAKICHKOLWIEK GWARANCJI (RĘKOJMIĘ RÓWNIEŻ WYŁĄCZA SIĘ), WYRAŹNYCH LUB DOMNIEMANYCH, A W SZCZEGÓLNOŚCI DOMNIEMANYCH GWARANCJI PRZYDATNOŚCI HANDLOWEJ, PRZYDATNOŚCI DO OKREŚLONEGO CELU ORAZ GWARANCJI, ŻE PUBLIKACJA TA NIE NARUSZA PRAW OSÓB TRZECICH. Ustawodawstwa niektórych krajów nie dopuszczają zastrzeżeń dotyczących gwarancji wyraźnych lub domniemanych w odniesieniu do pewnych transakcji; w takiej sytuacji powyższe zdanie nie ma zastosowania.

Informacje zawarte w niniejszej publikacji mogą zawierać nieścisłości techniczne lub błędy typograficzne. Informacje te są okresowo aktualizowane, a zmiany te zostaną uwzględnione w kolejnych wydaniach tej publikacji. IBM zastrzega sobie prawo do wprowadzania ulepszeń i/lub zmian w produktach i/lub programach opisanych w tej publikacji w dowolnym czasie, bez wcześniejszego powiadomienia.

Wszelkie wzmianki w tej publikacji na temat stron internetowych innych podmiotów zostały wprowadzone wyłącznie dla wygody użytkowników i w żadnym wypadku nie stanowią zachęty do ich odwiedzania. Materiały dostępne na tych stronach nie są częścią materiałów opracowanych dla tego produktu IBM, a użytkownik korzysta z nich na własną odpowiedzialność.

IBM ma prawo do używania i rozpowszechniania informacji przysłanych przez użytkownika w dowolny sposób, jaki uzna za właściwy, bez żadnych zobowiązań wobec ich autora.

Licencjobiorcy tego programu, którzy chcieliby uzyskać informacje na temat programu w celu: (i) wdrożenia wymiany informacji między niezależnie utworzonymi programami i innymi programami (łącznie

z tym opisywanym) oraz (ii) wspólnego wykorzystywania wymienianych informacji, powinni skontaktować się z:

IBM Corporation
Koordynator współdziałania oprogramowania, dział 49XA
3605 Autostrada 52 N
Rochester, MN 55901
U.S.A.

Informacje takie mogą być udostępnione, o ile spełnione zostaną odpowiednie warunki, w tym, w niektórych przypadkach, zostanie uiszczona stosowna opłata.

Licencjonowany program opisany w niniejszej publikacji oraz wszystkie inne licencjonowane materiały dostępne dla tego programu są dostarczane przez IBM na warunkach określonych w Umowie IBM z Klientem, Międzynarodowej Umowie Licencyjnej IBM na Program lub w innych podobnych umowach zawartych między IBM i użytkownikami.

Wszelkie dane dotyczące wydajności zostały zebrane w kontrolowanym środowisku. W związku z tym rezultaty uzyskane w innych środowiskach operacyjnych mogą się znacząco różnić. Niektóre pomiary mogły być dokonywane na systemach będących w fazie rozwoju i nie ma gwarancji, że pomiary wykonane na ogólnie dostępnych systemach dadzą takie same wyniki. Niektóre z pomiarów mogły być estymowane przez ekstrapolację. Rzeczywiste wyniki mogą być inne. Użytkownicy powinni we własnym zakresie sprawdzić odpowiednie dane dla ich środowiska.

Informacje dotyczące produktów innych niż produkty IBM pochodzą od dostawców tych produktów, z opublikowanych przez nich zapowiedzi lub innych powszechnie dostępnych źródeł. Firma IBM nie testowała tych produktów i nie może potwierdzić dokładności pomiarów wydajności, kompatybilności ani żadnych innych danych związanych z tymi produktami. Pytania dotyczące możliwości produktów innych podmiotów należy kierować do dostawców tych produktów.

Wszelkie stwierdzenia dotyczące przyszłych kierunków rozwoju i zamierzeń IBM mogą zostać zmienione lub wycofane bez powiadomienia.

Publikacja ta zawiera przykładowe dane i raporty używane w codziennych operacjach działalności gospodarczej. W celu kompleksowego ich zilustrowania, podane przykłady zawierają nazwiska osób prywatnych, nazwy przedsiębiorstw oraz nazwy produktów. Wszystkie te nazwy/nazwiska są fikcyjne i jakiekolwiek podobieństwo do istniejących nazw/nazwisk i adresów jest całkowicie przypadkowe.

LICENCJA W ZAKRESIE PRAW AUTORSKICH:

Niniejsza publikacja zawiera przykładowe aplikacje w kodzie źródłowym, ilustrujące techniki programowania w różnych systemach operacyjnych. Użytkownik może kopiować, modyfikować i dystrybuować te programy przykładowe w dowolnej formie bez uiszczania opłat na rzecz IBM, w celu projektowania, używania, sprzedaży lub dystrybucji aplikacji zgodnych z aplikacyjnym interfejsem programistycznym dla tego systemu operacyjnego, dla którego napisane zostały programy przykładowe. Programy przykładowe nie zostały gruntownie przetestowane. IBM nie może zatem gwarantować ani sugerować niezawodności, użyteczności i funkcjonalności tych programów.

W przypadku przeglądania niniejszych informacji w formie elektronicznej, zdjęcia i kolorowe ilustracje mogą nie być wyświetlane.

Informacje dotyczące interfejsu programistycznego

Informacje o interfejsie programistycznym, jeśli są dostępne, mają na celu pomóc w tworzeniu aplikacji do użycia z tym programem.

Podręcznik ten zawiera informacje na temat interfejsów programistycznych, które umożliwiają klientom pisanie programów w celu uzyskania dostępu do usług produktu WebSphere MQ.

Informacje te mogą również zawierać informacje na temat diagnostyki, modyfikacji i strojenia. Tego typu informacje są udostępniane jako pomoc przy debugowaniu aplikacji.

Ważne: Informacji o diagnostyce, modyfikacji i strojeniu nie należy używać jako interfejsu programistycznego, ponieważ mogą one ulec zmianie.

Znaki towarowe

IBM, logo IBM , ibm.comsą znakami towarowymi IBM Corporation zarejestrowanymi w wielu systemach prawnych na całym świecie. Aktualna lista znaków towarowych IBM dostępna jest w serwisie WWW IBM, w sekcji "Copyright and trademark information"(Informacje o prawach autorskich i znakach towarowych), pod adresemwww.ibm.com/legal/copytrade.shtml. Nazwy innych produktów lub usług mogą być znakami towarowymi IBM lub innych podmiotów.

Microsoft oraz Windows są znakami towarowymi firmy Microsoft Corporation w Stanach Zjednoczonych i/lub innych krajach.

UNIX jest zastrzeżonym znakiem towarowym The Open Group w Stanach Zjednoczonych i/lub w innych krajach.

Linux jest zastrzeżonym znakiem towarowym Linusa Torvaldsa w Stanach Zjednoczonych i/lub w innych krajach.

Ten produkt zawiera oprogramowanie opracowane przez Eclipse Project (https://www.eclipse.org/).

Java oraz wszystkie znaki towarowe i logo dotyczące języka Java są znakami towarowymi lub zastrzeżonymi znakami towarowymi Oracle i/lub przedsiębiorstw afiliowanych Oracle.

Numer pozycji:

(1P) P/N: